AN APPROACH TO DISTRIBUTED INFORMATION MANAGEMENT SYSTEMS FOR POWER SYSTEM APPLICATIONS

MA LI

A thesis submitted in partial fulfilment of the
requirements of the University of Abertay Dundee
for the Degree of Doctor of Philosophy

May 1995

DR. C S OZVEKEN
DIRECTOR OF STUDIES

To my beloved country - China

ABSTRACT

A distributed information management system combining a power system host computer and a number of local computer platforms will allow power system off-line engineers to share information from the host computer on their local computer platforms for power system analysis and planning. A common information management environment for the local computer platforms is the basis of such distributed systems. This work has concentrated on the development of a prototype of such a common local information management environment, called Power Shell, for PC platforms.

Power system management, operation, analysis and planning involve collecting, storing, communicating, evaluating and finally processing very large scale information and data sets. Power Shell has, therefore, been developed with the aim of enabling local PC platforms to match the capability of the host computer on managing one-line network diagram and device parameters in order to avoid data overflow when very large scale power systems are involved. A data flow scheduling technique has been developed and used for coping with the limitation of the local computer internal memory capacity. This technique consists of the concept of drawing boards, mapping and projection matrices, and the method of diagram information partitioning. Based on the capability of Power Shell in managing large scale information, which makes it possible to store very large scale power system static description data in local platforms, efficient communication in the distributed system can then be achieved by transmitting only the power system dynamic status data.

Other objectives achieved in the Power Shell development include the capability of accommodating different data formats for various power system application programs and allowing access to commercial software tools. This involves the use of a data dictionary technique for allowing flexible data format redefinition, and MS-Windows and PARADOX database techniques for implementing a standard user friendly interface and ensuring compatibility with commercial word processing and database systems.

As a running system, Power Shell has been tested using the data from some industrial power system networks. The test results have been satisfactory and demonstrated that a PC based common information management environment such as Power Shell is of significant benefit for power system studies and applications. It can assist power system analysis and planning for very large scale power system networks and can be used as a standard operation environment for various power system applications having different data formats, where the possibility of using other commercial software tools is also desirable.

ACKNOWLEDGEMENT

I would like to thank my supervisors Dr. C S Ozveren and Mr L Crowe for their help, advice and encouragement, and also Professor K L Lo at the University of Strathclyde as my external supervisor.

Thanks also to Dr P F Martin, head of the Department, and to Dr B Jefferies, former head of the Department, for the financial support for this work.

Thanks are also due to Mr K F Taylor and Mrs O Mariam, staff of Computer Centre, for the information about software development tools; and to Mr W Harper, Mr S Murray, Mr M Pacione, Mr W Butter and Mr J Mason, technicians of the Department, for their help with computer and laboratory equipment.

I would also like to thank my parents, my daughter and my wife for their understanding, constant support and encouragement.

GLOSSARY

ANDG:	Automated Network Diagram Generation
BDB:	Background Drawing Board.
Blackboard:	An ASCII file dynamically recording current job descriptions and formatted relevant data
BM:	BDB Mapping Matrix
Co-ordinate string:	Character string for presenting x-co-ordinate or y-co-ordinate
DD:	Drawing Diagram
DDE:	Data Dynamic Exchange
DIMS:	Distributed Information Management System
Drawing Board:	A logical plane with a co-ordinate system
LAN:	Local Area Network
LIME:	Local Information Management Environment
LOC:	Lines of code
JO:	Joint Object
One-line diagram:	Single line diagram
PM:	Projection Matrix
PSN:	Power system network
PTO:	Parameter Table Operation
SDB:	Sub Drawing Board
SQL:	Structured Query Language
Unit moving step:	viewport moving distance measure
VDB:	Viewport Drawing Board
VM:	Viewport Projection Matrix
$\oplus$:	Ordered appending of two co-ordinate strings, called co-ordinate string combination

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER 1

INTRODUCTION

In recent years, Distributed Information Management Systems (DIMS) have been investigated and developed to support the work of electrical power system off-line analysis and planning. This allows local computer users to have access to the same one-line diagrams and tabular displays as the operators of the host computer system, and also provides near real time information to the many groups outside of the Operations Department without adversely affecting real time system performance [TREF88]. Most DIMS approaches have been concerned with specific power system applications and the applicability of the DIMS systems developed has been limited by the computer internal memory capacity. To avoid the effort of designing specific DIMS with a fixed data structure for a certain application and also to enable local computer platforms to match the host computer on the capability of storing and managing diagrams, network topologies and parameters for large scale power systems, this research has concentrated on the investigation and development of a prototype Local Information Management Environment (LIME), named Power Shell, to form the basis of the DIMS, which can accommodate different data structures and its applicability is not restricted by the local computer internal memory capacity. This, therefore, can be used for various power system applications. The research work concerns also a communication strategy for allowing effective communication in the DIMS.

This thesis details the above research work. In this chapter, an overview of the requirements and problems in the development of the DIMS is given, the objective of this research and the work involved are outlined, the original contribution of this research is stated and the structure of this thesis is introduced.

1.1 Problem statement and solution requirements

A Distributed Information Management System (DIMS) is normally composed of a power system host computer (or server) and a number of local computer platforms. They are connected by a LAN and can exchange data with each other. It allows power system off-line engineers to share information stored in the host computer on their local computer platforms for power system analysis and planning. The software environments for information management on the host and local computers, called Local Information Management Environment (LIME), have to be compatible in order to provide users with a complete DIMS. Therefore the development of 1) the LIME that enables local computers to match the host computer on the capability of data management and 2) the strategy allowing an effective communication are the two key issues of this research.

One of the key problems in the development of such a LIME concerns its capability in information management for very large scale power systems. Normally a power system network is a complex and large system. The host computer of the power system network is required to record large amounts of data in order to manage information about network one-line diagram, topology, element parameters, running status, etc.. The data stored in the host computer will be required regularly by power system off-line engineers for their daily work concerning analysis, planning and calculations. Because a power system network works in its entirety, the information for off-line engineers is required to be that about a complete rather than part of network. This requires the DIMS to provide an operating environment that allows local computer users to have access to the same one-line diagrams, tabular displays and all other information about the power system network as the operators of the host computer system. The problem here concerns computer hardware conditions. For

economic reason, except the host computer that has to be one that has rich hardware resources and sufficient internal memory, other local computer platforms in the distributed computer network usually are composed of low cost PCs or workstations, i.e. the computers with comparatively limited internal memory capacities. So local users could be unable to access complete power system network information stored in the host computer just because of the lack of internal memory of local computer platforms. To overcome the problem caused by computer hardware conditions, appropriate software techniques enabling an effective and efficient use of the computer memory are significant in the development of the LIME of the DIMS. This means the LIME is required to, from software technique angle, enable local computer platforms to match the capacity of the host computer in managing information for very large scale power systems.

As a large amount of data has been involved in the DIMS operation, a problem has also arisen from the transmission speed allowed by today's communication techniques in dealing with real time information management. Generally the communication of a DIMS is achieved through a standard LAN. A serious transmission delay will be unavoidable when such a considerable amount of power system data has been involved. A strategy for reducing data redundancy in the transmission is, therefore, required in order to enable effective communications in the DIMS.

On the other hand, during the last few years, computer software industry has been undergoing a fundamental change since the appearance of the Windows techniques. This change is mainly driven by the increasing requirements of providing integrated, interactive and user-friendly operating environments. Meanwhile software development work has been further divided by product categories for basically two groups of developers. One group is composed of the professionals associated with the

software companies, often termed the commercial product providers. They supply the software market with integrated software operating environments, various programming languages and software development tools, such as X-Windows, MS-Windows, C and database systems. The other is a group of specific application program developers. They normally use the tools provided by the first group to develop specific application programs for the end users. This group of developers are required to have a deep knowledge of the application domains and be able to implement complex calculation programs. Considering user-friendly operations, they sometimes have to provide end users with not only the calculation program but also a specific operating environment for the calculation program. The latter process will normally require about 60% of the effort for the whole development work, with the calculation program itself requiring only about 40%. Therefore the development of such a specific environment for individual application programs has been a hard task for application program developers. It has also created the following two problems:

- Repeated efforts. Many efforts have been made to implement and improve individual environments that are only different from the data structures for the specific calculations or other applications but have identical basic operating frameworks.
- Non-standard user interface. These individual environments have been provided with various styles of user interface. Difficulties have arisen in the use of several different application programs, where the end user has to deal with many of these different operations.

These problems have specified the need for a common operating environment for managing power system network information which is capable of accommodating different data formats for most of the specific calculation and analysis programs.

1.2 Objective of this research

Considering the problems summarised above, the objective of this research is to investigate a common Distributed Information Management System (DIMS) to assist power system application programs in constructing one-line diagrams and storing network parameters. The system should be supported by an effective communication strategy and be able to serve very large scale power system applications that may have different data formats. It should provide users with the benefits as follows:

- to enable local platform users to access the database of a host computer through their LIME;
- to provide power system application developers with an open environment for embedding various application programs, share data managed by other LIMEs in the DIMS, and display calculation results;
- to provide power system application developers with a standard and universal environment for managing information in different data formats, and for exchanging data between different calculation programs;
- to provide power system off-line engineers with a software tool to bridge their daily information processing routine and commercial software tools such as word processors and database systems.

1.3 Outline of the work

To develop the LIME described in the above section, the work started with a literature survey for the analysis of the characteristics of the user operating environment required for power system application programs. This mainly includes the description for power system network topology, the amount of data to be managed, the real time aspects, the network diagram interface features and the various data formats to be

dealt with. A large amount of development work has been done following the survey and this includes:

- analysis and selection of the hardware environment for constructing the LIME;
- analysis and selection of the software tools for implementing the LIME;
- design of the DIMS framework and data location method to enable effective communication;
- feasibility analysis and data structure design for managing data and diagrams on PC platforms for very large scale power system networks;
- analysis of data relationships and design of the relational database for the DIMS;
- analysis of the requirement of the DIMS real time aspects and design of a data flow scheduling method;
- analysis, design and implementation of an object oriented project method for power system project management;
- analysis, design and implementation of a data management method allowing a flexible data structure; and
- analysis, design and implementation of a diagram drawing method on PC platforms for very large scale power system networks.

The outcome of the above work is a prototype LIME, called Power Shell, of the DIMS. Power Shell has been written in a combination of MS-Windows, C and Paradox Engine. The Power Shell system has been tested and evaluated using data from industrial power system examples and the results have been satisfactory for the achievement of the DIMS research objective. The concepts, methods and techniques involved in this research work have been written in papers presented at a number of conferences [MALI92] [MALI93c] [MALI94a] [MALI94b] [MALI95], where the Power Shell system itself has also been demonstrated. The demonstrations have been of interest to attendees from both academic and industrial circles.

1.4 Contributions of this research

This research has investigated and developed an Information Management Environment, called Power Shell, forming the basis of a future Distributed Information Management System for various power system applications.

There are mainly three contributions of this research. Firstly, a data flow scheduling technique has been developed and used in the Power Shell implementation for removing the limitation of PC internal memory on handling large scale power systems. This technique involves the concept of drawing boards and projection matrices as well the method for partitioning diagram information. Therefore, in the distributed system using Power Shell as a local information management environment can enable those local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and parameters for large scale power systems. Usually the problems of data overflow have happened because of the lack of capacity of the PC memory.

Secondly, a concept of only transmitting dynamic data has been proposed for efficient communication in the distributed system. Supported by the static data, i.e. diagrams, network topologies and parameters, of local PC platforms, only the data of the current plant status such as dynamic data, needs to be transmitted when the user asks for information sharing from the host computer or other local PCs. Thus it can reduce the data redundancy in real-time communication required by the distributed system.

Thirdly, Power Shell has provided an integrated graphical and database management support for assisting various power system application programs in constructing one-line diagrams and storing network parameters. The use of Power Shell will greatly

assist power system application program developers in the design and implementation of a professional user interface and data management environment. The main features and benefits of the Power Shell system include:

- **large scale power system modelling support**

  Power Shell allows users to construct one-line diagrams of large scale power system networks and record corresponding parameters. For Power Shell the restriction on the scale of power system depends only on the capacity of the PC hard disks rather than the internal memories through the use of data scheduling and database techniques.

- **automated network topology description generation**

  Power Shell can automatically produce power system network topology descriptions based on network one-line diagrams. The users of Power Shell do not require to input literal descriptions of the power system network topology initially. When the users draw a power system diagram through the Power Shell diagram interface, Power Shell automatically translates the power system element relationships presented by the diagram into literal descriptions and writes them to the database table. These descriptions will be prefixed to the corresponding items in the ASCII files for data exchanges.

- **open system**

  Power Shell supports data access to other software programs in four ways:

  - Users can directly open the Power Shell database tables in a relational database environment such as PARADOX and dBASE IV for data ordering, indexing and analysis.

- Power Shell can transfer the data from the database tables into standard ASCII files that can be further processed by users' application programs. These ASCII files can also be transferred to different kinds of computer platforms such as APOLLO and SUN workstations. The ASCII files produced by Power Shell contain the following information of a power system network:

  * network topology;

  * circuit breaker status;

  * data and parameters of network elements.

- The data output of user application programs or on-line data acquisition programs can be read back to the database of Power Shell, and can then be displayed on the screen diagram. The read-back data of a power system element can also indicate the element current status by keeping or changing its display style in the screen diagram.
- The screen diagrams of power systems produced by Power Shell can be copied to existing commercial software tools such as Microsoft Word and Microsoft Write in order to generate reports.

- **flexible data formatting**

  To accommodate various input ASCII file formats required by power system application programs, Power Shell allows flexible data formatting based on user requests. In the ASCII files produced by Power Shell for data exchanges, each data record is composed of two parts: prefix and parameter. The prefix part is for describing relevant element relations. The data fields and the order of the fields in the prefix part can be selected by users through the Options Dialog box (Fig1.1).

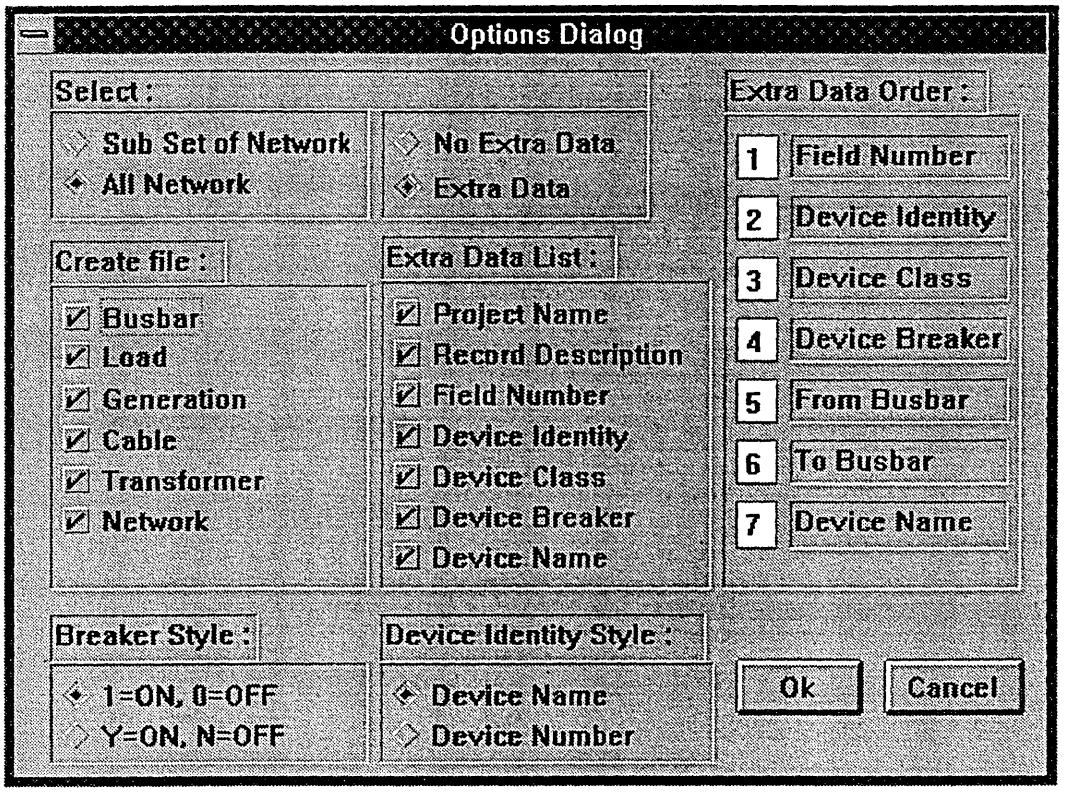


Fig 1.1 Power Shell data export Options Dialog box

For the parameter part, a default database data structure is given by Power Shell and it can be redefined by users(Fig 1.2). For each element, users are allowed to define up to 80 data fields. In Power Shell system operation the database tables and then the ASCII files are formatted automatically to match the specified data structures. Therefore the format of the ASCII files to be produced can be completely controlled by Power Shell users.

with the development of such data management systems and indicates the major existing problems. It concentrates on illustrating the needs of a universal information management environment, for power system applications, which can accommodate different data formats and is able to manage information for very large scale power system networks.

Chapter 3 discusses the development of the framework of a Distributed Information Management System(DIMS). This includes consideration of the DIMS real-time and graphical characteristics and requirements, the analysis of current LAN techniques and DIMS data categories, and the design of a data distribution method for enabling effective communication in the DIMS. This framework has also specified the requirements and features of its Local Information Management Environment (LIME).

In Chapter 4, based on the DIMS framework, the database system for the LIME is analysed and designed. This involves the analysis of the power system data relationships, the design of database data table structures and drawing diagram data structures. The database system design is the key to enable the LIME to have a high capability for data management.

In Chapter 5, work is continued on the development of drawing methods for very large power system network diagrams. The methods for two different drawing styles, automated drawing and CAD based drawing, are discussed and designed. This involves respectively a recursive method for managing the use of buffers and graphical information partitioning techniques. The latter has been incorporated in the LIME, whilst the former demonstrated in a separate prototype for comparison. Both methods have been implemented with success in dealing with one-line diagrams for very large power system networks.

one-line diagram for most operations involving network elements. (3) Make the one-line diagram reflect the current state of the system. In short, what you see must be what you get. [CHAN90]

Chan's description has summarised the main features of the interactive graphics interface. The concept of the User-Conceptual Model has been accepted and used in most approaches in this area. Comparatively some earlier approaches to automated generation of one-line diagram have been less helpful in graphical interaction. The only advantage of the automation is to save the time for constructing one-line diagrams. Therefore the LIME development work of this research mainly concentrates on the interactive graphics interface, although the automated method is investigated for comparison.

There are still some limitations on the approaches to interactive graphics interface. For the PC based systems such as the ASEA BROWN BOVERI (ABB) system, its capability for data management has been limited to 800 buses and 2000 elements [CHAN90]. The main reason was the limitation of the PC internal memory. To provide a solution to this problem is one of the contributions of this research. In summary the LIME to be developed by this research work should provide the user with an interactive graphics interface based on the User-Conceptual Model, and also remove memory limitations to enable local PC platforms to match the host computer on the capability of managing diagrams and parameters for large scale power system networks.

2.3 Open systems

With the increasing spread of software to various areas and the rapid updating of software development techniques, more and more specific power system application programs have been developed. Normally an application program will require historical or current power system running status data as input for the specific analysis and calculations. Sometimes it will also require data exchange with other power system applications. Therefore the benefits of an open operation environment concept, where application domain engineers can embed their calculation program into the environment and share the data provided by the environment, are becoming increasingly established and accepted. This has been of interest to quite a few researchers [VADA93] [KLEN92] [SASS92] [BRIT92].

Within these approaches, Vadari has given a clear description of open systems[VADA93]. The description has been made from a 'system' point of view, which involves open system hardware, operating system environment, user interface and database. The following quotation from Vadari is the answer to "WHAT ARE OPEN SYSTEMS".

One way to judge if a system is open is to look at the system from the users' point of view. From the users' perspectives, applications and ability to access data determines openness. Users would like their application to operate the same, independent of hardware, operating system, user interface or networks. For example, a truly open spread sheet application would run on any hardware, with any spread sheet program, and user interface, and would look and behave the same in all cases. It would also interface to any word processor, or database.[VADA93]

It should be clarified that what Vadari described is an ideal open system, which could not be completely achieved with the current computer hardware conditions. Currently the CPU used for different kinds of computers requires a different executable code. Any high level computer language may allow programming in identical source code on various computer platforms, but the executable code compiled for different computer CPUs is different. Therefore the expectation that software programs could be independent of hardware requires a fundamental change in the hardware industry. However, except for the hardware limitation, for a certain kind of computer (e.g. PC) it is possible to construct an open system by means of software. The hardware specified open systems would better suit current computer application techniques.

The requirement for the user interface of an open system for power systems has also been discussed by Vadari [VADA93]. Taking for example the EMS (Energy Management System) dispatcher, this could include any one of the following:

"1) Interacting with the EMS (typically a mainframe-type hardware implementation) to perform standard control and dispatching functions. This would involve the use of the user interface provided by the vendor and their operating practices.
2) Interfacing with the energy accounting system. This would typically be a PC-type system with the energy accounting data being resident in a relational type data base, and would require the user to learn the use of that new tool.
3) Writing daily/weekly reports. This would also typically involve a PC-type system where the operator would have to use a word processing system."[VADA93]

From Vadari's discussion it is clear that the Open System User Interface is not a simple utility, it is actually more like the LIME of the DIMS. This LIME should be able to interface with, at least:

- users,
- database systems,
- application programs,
- word processing systems.

Besides, it should also provide power system diagram interface and data management utilities. All these are the characteristics necessary for the LIME to be an open system.

From a technical point of view, in proceeding to an open system, the LIME should be implemented using industrial standard software techniques such as MS-Windows techniques. This will, firstly, enable a standard user operating environment and then allow data exchange with word processing programs via MS-Windows. The use of standard relational database techniques will also enable the LIME database data tables to be accessed from other commercial relational database systems or Structured Query Language (SQL).

Another important requirement for an open system is the development of a data exchange interface for LIME to exchange data with power system application programs. Difficulties here are mainly concerned with the variety of data formats used by these application programs. To solve the problem there have been two different kinds of approaches. One is to conditionally open the system. The basic idea is to open the LIME data format and allow application programs to share the data. However in case the data format provided by the LIME does not match the requirement of the application program, then the LIME data could not be shared unless a change in the data format of the application program has been made. Another is to provide the LIME with a full set of power system parameters to allow the user to specify the required ones in a certain format[GONE84]. This may be more convenient than the first approach, but the problem is how to define the 'full set'. A parameter set

could be called a full set today but might not be tomorrow since the updating of the calculation methods or application techniques could require more parameters or other factors to be involved. Therefore the open systems proposed by this approach have been limited to a predefined parameter set.

To remove the above limitations, the development of the LIME should consider the changeability of its data format. One practical and effective way is to provide the LIME with data format definition facilities and allow the users to update the LIME data format to match the requirements of the application programs. This could be achieved by using the relational database data dictionary techniques. Thus the LIME could be an open system allowing users to organise applications and data formats.

2.4 Importance of an information management environment for power system application programs

The power system analysis and calculation programs require large amounts of historical data, which includes:

- power system network topology description;
- power system parameters;
- power system running status records.

The data required as the input for these application programs is normally recorded in a certain format in ASCII files. In application program operation, the program will first load in the data to the internal memory, and then start the analysis and calculation process. The results produced by the application program will also be written to a corresponding ASCII file in the predefined format. The resulting ASCII file can be stored on disks or printed out.

To produce such input ASCII files for the application programs to read is very time consuming and requires considerable effort. Also the literature description of the network topology is usually difficult to make sense of even for power system engineers. To overcome these difficulties, the Information Management Environment (IME) for power systems has been desirable. The IME is basically required to have two major parts: Database and a Power system network diagram interface. Database is for recording and managing all the information on the power system networks. This information can be provided directly in the format required by the application programs. Diagram interface is for drawing power system network one-line diagram to provide the user with a visual display of the network topology, allowing the user to access power system element parameters via the diagram, and making the one-line diagram reflect the current state of the power system or application results.

The question is how much effort will be required to implement such an IME. It may be difficult to give a direct answer but the following example, however, could be of some help. At the IEE Power Division Colloquium 92, Mr Parton from ICL described the interactive useability features of an interactive graphics based power system analysis program for system planning, called DINIS, and then the effort required. This has been quoted as follows:

....., and the key features needed so far have been found to be:-

a) Automatic and untouchable connectivity tables derived directly from the drawing process.

b) Optional node and line identifiers (i.e., non-mandatory and which can be duplicated without introducing topology problems).

c) Full network and file flexibility for planning options.

d) Automatic positioning of all output results.

e) Visual network compare and update (file merge) facilities.

g) Easy data handling:-.....

These requirements result for example in DINIS currently consisting of 12,000 lines of classic network analysis code, 80,000 lines of engineering user interface code, 50,000 lines of data base and network handing plus a final 20,000 covering general IT technology to keep pace with fast changing IT hardware and developing open system S/W standards.[GADS92]

From the above example, it could be seen that a power system application program might require 80% effort for its IME. This has identified the importance of the IME part in an application program. Therefore the significance of a common IME which is able to accommodate different data formats and thereby serve most power system application programs has been clarified. Such a common IME will release engineers and researchers, whose target products are power system application software programs, from the restrictions in the design and implementation of a professional user interface and data management environment.

2.5 ASCII file data format requirements for power system application programs

Currently various input ASCII file data formats have been used in power system application programs by individual companies, researchers and engineers since there is no existing standard format. The design of the data formats could be based on individual factors, such as the developer's experience or preference, application target, reducing data redundancy, easy to construct calculation matrices, special parameters, optimisation of internal memory management, readability, or even just for the purpose

of having a special style. For example, the difference in the data formats for power system element Line used by PSS/E, ABB and Power Tool Box is shown as follows:

Branch Data (PSS/E)

I, J, CKT, R, X, B, RATEA, RATEB, RATEC, RATIO, ANGLE, GI, BI, GJ, BJ, ST

Where

I = branch "from bus" number. For a transformer, this bus is the tapped side bus.

J = branch "to bus" number. For a transformer, this bus is the impedance side bus. J is entered as a negative number to designate it as the metered end for area interchange and zone calculations. Otherwise, bus I is assumed to be the metered end.

CKT = two character upper case alphanumeric branch circuit identifier; the first character of CKT must not be an ampersand. It is strongly recommended that single circuit branches be designated as having the circuit identifier "1". CKT = 1 by default.

..... [POWE89]

Line data (ABB)

Element name, Active, Line, Bus name 1, Bus name 2, Un, Sr, Is, R1, X1, Cy, Ro, Xo, Cyo, Temp c, Length, Pu-Values

Where:

Active: Flag, which shows whether the element is in operation or not.

Line: Flag, which shows whether the element is a line or bus coupler.

Bus name 1: Name of nodes at the beginning of the line.

Bus name 2: Name of nodes at the end of the line.

..... [ABB93]

Line data (Power System Tool Box)

From bus, To bus, resistance, reactance, line charging, tap ratio, phase shifter angle.

Where:

From bus: Number of bus at the beginning of the line.

To bus : Number of bus at the end of the line

..... [CHER94]

This variety of data format has been a major challenge to the common IME for power systems. First of all, it is difficult to obtain all the input ASCII file data formats that are currently being used in the world. Secondly, it is impossible to predict the data formats that may be used later. Therefore, as mentioned previously, the best way is by allowing users to update the data formats supported by IME to match their specific requirements.

2.6 Database techniques

Although, as is common knowledge, database techniques have to be involved in the development of an Information Management Environment (IME), no specific requirements for database techniques in power system IME had been identified until the IEE Power Division Colloquium in November 1994.

One discussion on database features was made by Mr Hainsworth from Ferranti Syseca Ltd: "*A feature of the data base design is that it allows the customer significant freedom in specifying the structure of the data base. It also accommodates updates to the table and data base structure without the need for software modification. These factors cause a number of implementation problems for the Data base Generator software which has to map the user defined data base structure onto a*

strictly defined PSS/O data format. The solution to these problems has been achieved through the use of metadata in the RDB, ie tables that describe the use and content of the customer specified tables" [HAIN94].

Mr Parton from ICL has also indicated the requirement on database design for power system information management, as quoted below:

From the preceding sections, various conclusions can be drawn as follows;

a) The first essential is that any data base needs to be extremely flexible in itself, or have a flexible interface with other corporate systems. Local data base will constantly change as new requirements and technical options emerge.

b) The next essential is to ensure that any central "pot" only holds genuine data of a corporate need, thereby giving utmost freedom of manoeuvre to other responsible specialist departments.

c) Distributed databases are needed that are linked together so that all common data is only entered once by one responsible dept. and immediately available to others. The agreement on the primary owner of key data is a significant management decision to be made.

.....

f) To aid the easy and prompt traffic between different data systems, it would appear attractive to interpose a universal data base system, probably using relational database tools, to act as a general purpose gateway between all other systems. Not only can this ensure a common point of information access, but enables any major new system developments to immediately become part of the data system, and hence enable technical users to extract any data for new technical programs or queries. [PART 94]

From the above discussions, it is clear that a flexible data structure has been recognised as significant for the design of a database system for managing power system information at this recent colloquium. Actually this has also demonstrated the correctness and suitability of the database techniques used in the development work of the LIME by this research, which started in November 1992.

The use of database techniques in the development of the LIME has been concerned with the requirement previously discussed, i.e. to enable the LIME to accommodate different data formats. Therefore the relational database data dictionary techniques has been used to allow the user to redefine the data structure when necessary. The development of the LIME was completed in April 1994 with success in accommodating flexible data structures. Users can redefine the data structure of the LIME database data tables, which will not destroy the data index. After the redefinition the LIME will automatically match the updated data structure for all services and no alteration to any sub programs is required. This has been further detailed in Chapter 4.

2.7 Management of information for very large scale power systems

It was indicated by Trefny, "Most importantly, operators at distribution zone offices and power plant control rooms have access to the same one-line diagrams and tabular displays as the system operators" [TREF88]. This should be considered in the development of a DIMS for power system applications. Actually the issue emerging from this requirement is to enable local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and parameters for power systems. The difficulties arising concern the power system size, i.e. how

many elements need to be dealt with. This size problem for data management has been discussed by Parton [PART94], as quoted in the following paragraphs.

The basic size of distribution network within one PLC is substantial, for example, secondary (11KV/415V) substations can number up to 40,000, the 11KV network may have up to 1 million supporting wood poles (all expecting to be uniquely identified), and for network analysis studies, inter-connected networks of several thousand active nodes can be quickly generated. Indeed, solutions of up to 100,000 nodes have been reported.

Apart from the size of the network itself, the supporting data needed for detailed planning studies can be equally large. For example, some users are finding the need for over 1,000 different line codes, with all possible combinations of different conductor types in each phase and neutral.

When considering a normal three phase transformer itself, the current data fields have grown to over 100 for accurate modelling, plus a 6 by 6 primitive impedance and a 9 by 9 connectivity matrix for unbalanced analysis. When studying banked single phase transformers such as open Wye/open Delta, the GE (USA) Distribution Transformer Manual lists over 40 recommended standard connections.

Furthermore, when considering digitised map background requirements and other associated (GIS) data, current technology is indicating over 1 Mbyte for each 1:10,000 stored map, so that putting all the information together is generating data storage needs in the multi Giga byte range.

Finally, it must be accepted that the growth of this data will never stop. For example, current considerations of demand side management could lead to an intelligent meter in every house, ie over 1 million complex units within the next 10 years all needing and generating unique additional data.

Thus apart from the sheer data volumes involved, even by modern computer standards, the complexity of the data management task itself becomes the prime consideration.

To solve this complex data management problem has to be considered in the development of the LIME expected to suit very large scale power system networks. To provide a solution to this problem is one of the contributions of this research, which involves the development and use of appropriate data flow scheduling methods discussed in Chapter 5.

2.8 Summary

In the above survey the useful techniques applied in and limitations of relevant approaches have been discussed with each issue considered respectively. In addition related existing commercial software packages have also been investigated. These are listed in Table 2.1. In summary the relevant and important techniques that should be used in the development of the LIME and thereby DIMS include:

- Interactive graphics interface techniques;
- Open system techniques;
- Relational database techniques.

The key issues to be dealt with in the LIME development are:

- to enable local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and element parameters for very large scale power systems;
- to provide the user with database data structure redefinition facilities, thereby allowing the accommodation of various power system applications that require different input ASCII file data formats;
- to be compatible with commercial word processing and database systems to benefit users by giving access to different software tools.

All these have formed the initial considerations in the LIME development carried out by this research work. The techniques mentioned above are discussed further in relevant later chapters and are used in the development of a LIME meeting all the key requirements summarised here.

Producer	Package
ABB Netcom ltd	POSCODAM (Power System Computation and Data Management)
Cherry Tree Scientific Software	Power System Stability Tool Box
•EDSA Micro Corporation	Power 2000
ICL	DINIS Electricity
Operation Technology and Power System group	ETAP (Electrical Transient Analyzer program)
Power Instrumentation & Control	X-ECUTIVE
Power Technologies, Inc.	PSS/E
Roettger Engineering Software Corporation	EZ-Flow
Scott & Scott Systems, Inc.	SLD (Single Line Diagrams) DIG (Database Via Digitizer) CWOS (Construction Work Order System)
SKM System Analysis, Inc.	Power Tools Software
UMIST, Electrical Energy and Power System Group	IPSA9, IPSA10

Table 2.1 Relevant commercial software packages

CHAPTER 3

FRAMEWORK DESIGN FOR A DISTRIBUTED INFORMATION MANAGEMENT SYSTEM

The Distributed Information Management System (DIMS) discussed here is intended to manage one-line diagrams and element parameters for very large scale power system networks. Because of the real-time aspect and graphical requirements, data storage and communication are the two key technical issues in the design of the framework of such a DIMS. A method of categorising power system data and constructing a distributed database system has been developed by this research, in order to provide a practical way of dealing with the above issues under the present computer hardware conditions and communication techniques. This is one of the important elements that form the basis of the DIMS.

3.1 Basic components and tasks of DIMS

Generally a DIMS is composed of two parts, one of hardware and the other of software. The hardware of a DIMS combines several computers with a Local Area Network (LAN) communication bus. One of these DIMS computers, which has a comparatively large hardware capacity, is used as the Host computer or Server, while others are the local platforms for local users. The software of a DIMS consists of a Local Information Management Environment(LIME) for the local platforms and the corresponding communication program. For example, suppose the DIMS contains n local workstations presented by LIME(i), i =1,2,..., n; then any LIME(i) can share data with LIME(j), $j \neq i$, if the user of LIME(i) is allowed to access all the data in the DIMS.

For the DIMS for very large scale power system networks, the basic tasks in consideration of system readability and understandability are to provide one-line diagrams, network topology descriptions, element parameters and other relevant data . Therefore each LIME of the DIMS is required to record the following data categories D_i, i=1,2,...,8, for one power system element:

D_1: relations with other elements;

D_2: graphical description data;

D_3: element class;

D_4: element name;

D_5: element characteristic parameter (p) p=1,2,...;

D_6: element breaker status;

D_7: element current running status (or historical running status);

D_8: analysis, calculation result or measurement data.

Furthermore, since all the elements of a power system network are related to each other, it is important for the DIMS and the associated LIME to be able to provide the complete power system network topology with relevant data rather than segmented or simplified information.

3.2 Requirements on data storage and communication

Normally a professional power system network(PSN) contains thousands of elements, E_i, i=1,2,... . If each element needs to be described by the eight data categories mentioned above, then the DIMS and LIME should be able to support the storage and management of the following data in case the element amount is not less than one thousand:

PSN: { E_i, i=1,2,...n, n >1000);

E_i: {D_{1i}, D_{2i}, D_{3i}, D_{4i}, D_{5i}, D_{6i}, D_{7i}, D_{8i} }.

The storage space required for describing a data field of an element is related to the data category. In general, the data categories of D_1, D_3, D_4, D_6 require relatively less bytes space than the categories of D_2, D_5, D_7, D_8 . The actual space required for the data storage depends on the data structure design and data length. Suppose the average space required for a data field of an element record is m bytes, then each element description requires 8×m bytes storage space. When m >100, for a professional power system network with more than one thousand elements, the storage space required to

record all the elements to the database will be more than $8\times100\times1000=$ 800,000 bytes.

With the above estimate of the data and storage space required for a professional network, the question now is: can the communication involving such large amounts of data in the DIMS be supported by the present computer communication techniques. The current computer communication data transmission speed is shown in Table 3.1.

Bit (second)	1200	2400	4800	9600	19200
Bytes (second)	150	300	600	1200	2400
Transfer by telephone line	√	√			
Transfer by local area network				√	√

Table 3.1 Current communication data transmission speed

In the computer LAN, the communication speed basically depends on the characteristics of the network cables. On the other hand, as the communication speed increases the data transfer error will be correspondingly increased. In general, the transfer speed of 9600 baud rate is a popular choice. Suppose the host system of the DIMS has stored all the information of a power system network(PSN) with a data amount of m bytes and $m \geq 800{,}000$, and the user of one of the LIME requires all the above data for a PSN simulation program. Then the time needed to transfer all the PSN data will be, at least, equal to T where:

$$T = \frac{800000(byte)}{1200(byte/\text{sec.})} = 666(\text{sec.}) \approx 11(\text{min.})$$

The time required for transmission will be much longer for a PSN contains much more than a thousand elements. This transfer speed, even in the above case, is clearly unacceptable for a real time information system. This means that because of the

limitation of today's communication techniques on the transfer of a large amount of data, a delay on data transfer may occur in a distributed system where all the PSN data is stored only in the central database system. There is also another reason for the delay. When several LIMEs of a DIMS ask to share data at the same time, the contention can also cause delay on data transfer. Although this may be dealt with by a scheduling method such as 'first come first served' and 'time division', the delay is always related to the number of the LIMEs involved in the contention.

Communication efficiency is one of the topics of the computer LAN techniques. The interest of this research concerning this issue is to raise communication efficiency by reasonably reducing data redundancy rather than developing a new LAN technique.

3.3 Data distribution method design

To solve the problems in transferring a large amount of data in the DIMS by properly reducing redundant data, it is necessary to analyse the PSN data characteristics in the DIMS communication. For a PSN, if on-line acquisition programs take 'second' as the unit of its acquisition cycle, then some of the PSN data categories listed in section 3.1 may not be effected by time. These are:

D_1: relations with other elements;

D_2: graphical description data;

D_3: element class;

D_4: element name;

D_5: element characteristic parameter (p) p=1,2,...;

D_6: element breaker status.

The other two data categories are closely related to the real-time status. These are:

D_7: element current running status (or historical running status);

D_8: analysis, calculation result or measurement data, where D_8 depends upon D_i, i=1,3,5,7 and time t.

According to the time-related characteristics of the PSN data, the data can be grouped as static and dynamic data, i.e. data category D_1-D_6 is static data and D_7,D_8 is dynamic data. Based on this static-dynamic concept, a new method for data distribution has been designed by this research for reducing data redundancy in DIMS communication. The basic idea of the method, in addition to the central database storing all the data of the PSN on the host computer, is to set a local database for each LIME to store the above static data. Thus the communication within the DIMS can concentrate mainly on the dynamic data, basically D_7, and the amount of data transfer has been reduced.

Based on this method for data distribution in the DIMS communication process, the PSN data is utilised in two different ways, dynamic data being updated all the time and static data being updated only if the topology of the PSN has been changed. Fig. 3.1 and Fig. 3.2 illustrate the dynamic data and static data flowcharts respectively.

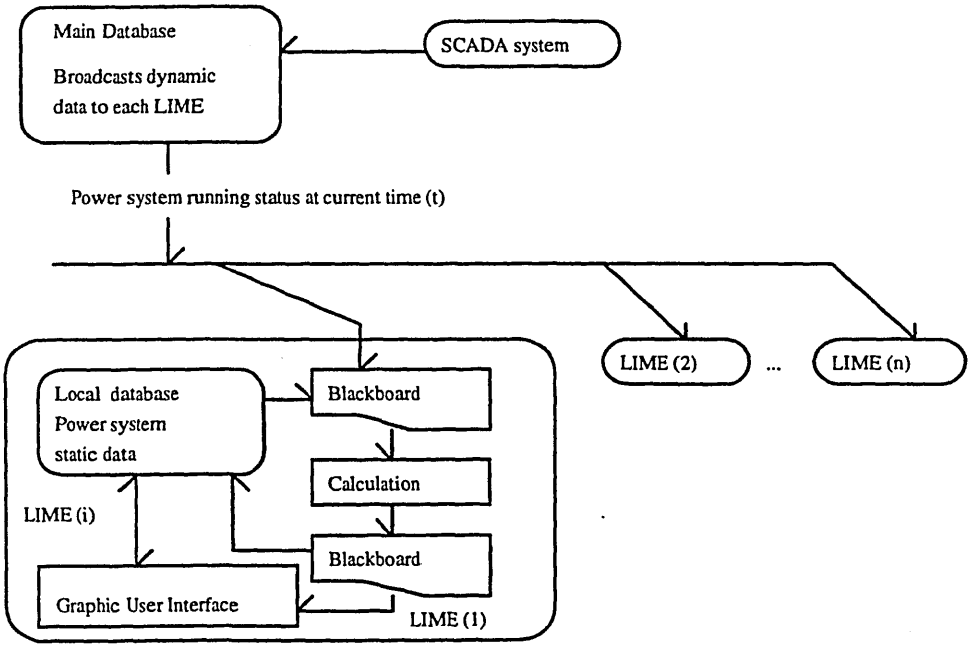


Fig.3.1 Dynamic data flowchart in DIMS

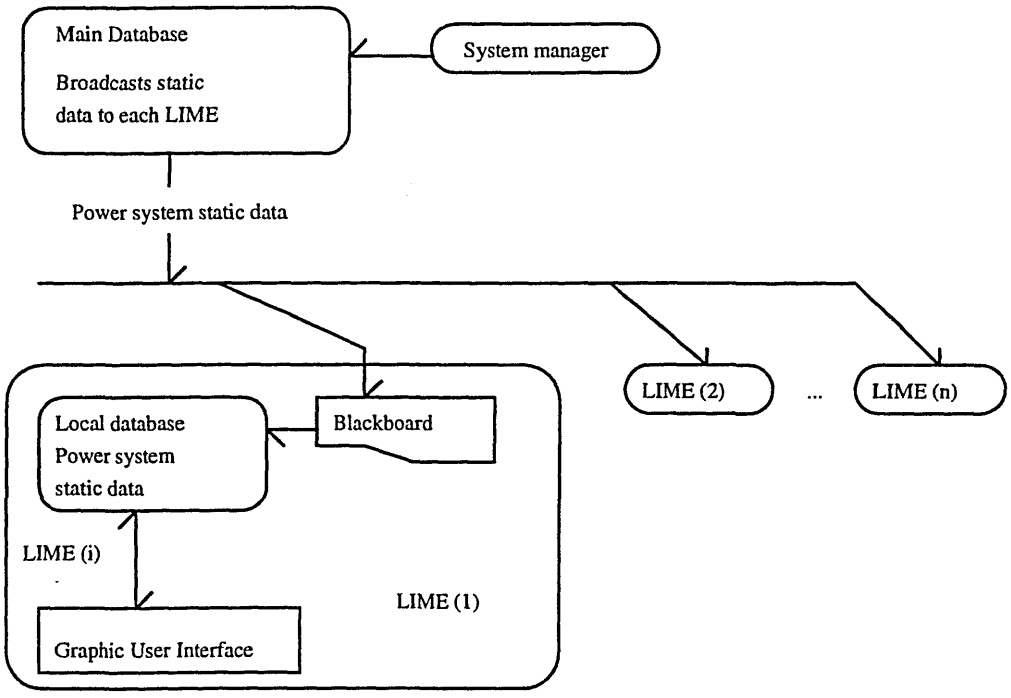


Fig.3.2 Static data flowchart in DIMS

It can be seen from Fig.3.1 and Fig.3.2 that although the paths for transferring dynamic and static data are identical, there is a difference in the relationship of data transfer with time. The dynamic data describing the PSN running status is stored in the main database and is continually updated. For the user of the LIME to get the data of current running status of the PSN timely, the main database can send the dynamic data to the LIME every few seconds. Since no static data is involved in this communication, the transfer time required can be effectively reduced from T, mentioned in section 3.2, to T'.

$$T' = \frac{D_7}{D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + D_7} T$$

The communication involving static data need not be very frequent, because the PSN topology changing cycle is much larger than the load flow changing cycle, and normally may be days, weeks or even months.

On the other hand, this distributed data storage has caused some data redundancy since the static data of the PSN has been stored not only in the main database on the host computer but also in each LIME. The question is whether this redundancy on data storage is worthwhile for the purpose of reducing communication costs. In general, the data required for, and application programs related to, a PSN can be illustrated in Fig. 3.3.

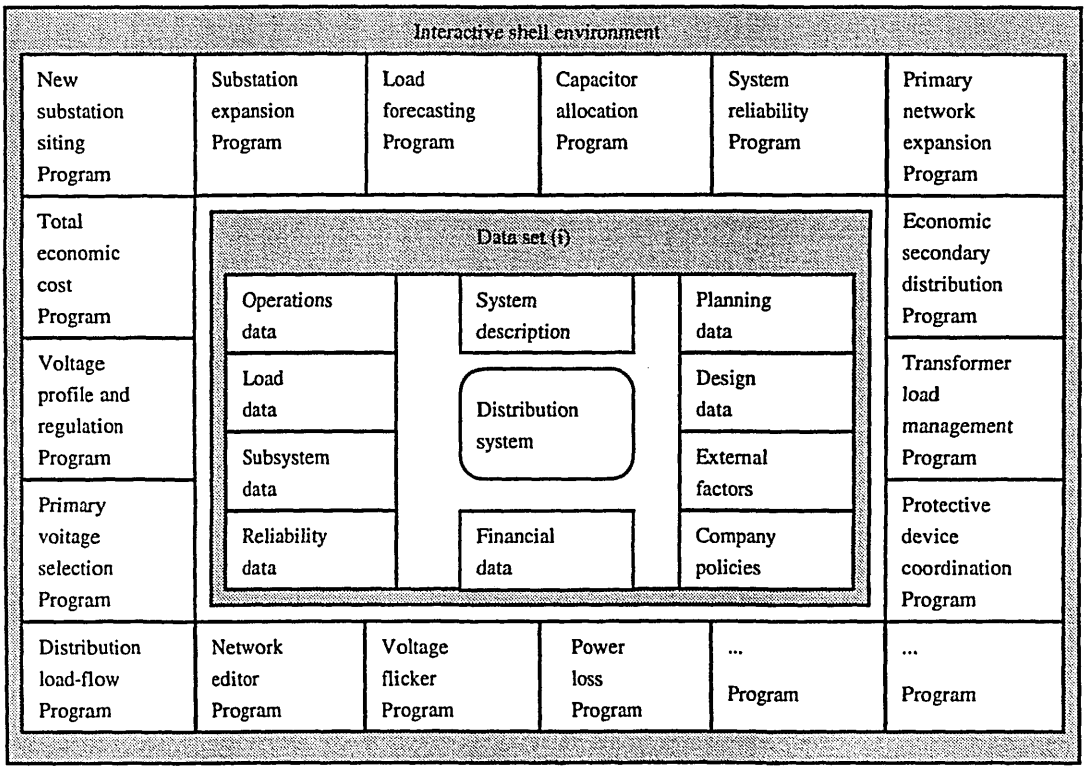


Fig. 3.3 Data set table

Considering the specific tasks for individual engineers involved in power system management, only the relevant data and the application programs need to be stored in the LIMEs. Fig. 3.4 depicts a LIME with a specific data set (*i*) (i=1,2,....;) that

depends on the data of the PSN topology and one-line diagram, and the specified tasks.

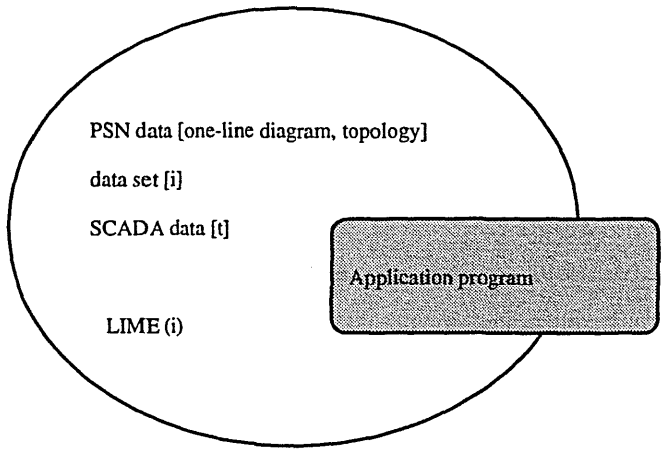


Fig. 3.4 A LIME for power system engineers

Therefore the data distribution in the main database and in the LIME databases in a DIMS with n LIMEs can be further organised as:

Main database { PSN data [one-line diagram, topology], SCADA data},

Local database 1 {PSN data[one-line diagram, topology], data set[1], SCADA data[t]},

:

Local database i {PSN data[one-line diagram, topology], data set[i], SCADA data[t]},

:

Local database n {PSN data[one-line diagram, topology], data set[n], SCADA data[t]};

where SCADA data [t] means the SCADA data at time t.

It is obvious that this distributed data storage has caused data redundancy, where

Redundant Data = {data[one-line diagram, topology], SCADA data[t]};

i.e. the data is repeatedly stored in the main database and all the local databases. Since the storage space (bytes) required for a PSN topology and diagram is much larger than that for the current SCADA data, i.e.

Bytes for PSN data[diagram, topology] >> bytes for SCADA data[t]

the time required for transferring the static data relating to the PSN is much longer than that for transferring current SCADA data. So the data distribution method proposed, by storing the PSN description static data and transferring only SCADA dynamic data, has reduced the overall data amount in the DIMS communication and raised the system efficiency for meeting real-time characteristic requirements. On the other hand, the redundancy on data storage will not be a problem for present PC platforms having about 600 Mbyte hard disk storage space.

3.4 Essential issues for LIME development

Based on the data distribution method discussed in section 3.3, there are two essential issues in the development of the LIME for the DIMS.

Firstly, the LIME must enable local computer platforms to match the host computer in its capability for storing and managing one-line diagrams, network topologies and parameters for large scale power systems. This will require support on data management, and in general the relational database techniques may provide a solution.

Secondly, the LIME must be able to manage different data sets for individual local workstations. This requires the LIME to accommodate various data structures and therefore a user-definable data structure construction should be considered.

The requirements identified here have confirmed the first two of the three key issues summarised in Chapter 2, while the third concerns the compatibility with commercial software tools. All these have been considered in the LIME development.

3.5 Summary

In this chapter, the framework for a DIMS, aimed to serve very large scale power systems, has been discussed and designed. This includes mainly the design of a data distribution method based on the use of a distributed database system for allowing effective communication in the DIMS, and the specification of the requirements for a LIME of the DIMS. The requirements focus on the capability of the LIME in managing information for very large scale power system and in accommodating different data formats for various power system applications.

Based on the work described in this chapter, the development of the DIMS will now concentrate on the development of a LIME that meets all the requirements specified in this chapter and also that in Chapter 2. The LIME development work is detailed in the following chapters.

CHAPTER 4

DESIGN OF THE DATA STRUCTURE FOR A LOCAL INFORMATION MANAGEMENT ENVIRONMENT

Data structure analysis and design is one of the key steps in the development of a Local Information Management Environment (LIME); it has to be taken before any coding work. The objective of the analysis and design is to build a suitable and complete data structure for the system being developed. The data structure should be able to provide a complete description to the data involved, which could include a set of tables, arrays and matrices for the static descriptions on

- time,
- sequence,
- status,
- parameter,
- diagram,
- data relationships,
- data scheduling key,
- file structure,
- processing report.

Based on these static descriptions the dynamic processing of the static data thus described can be driven by the application computer programs. Therefore the data structure analysis and design is the basis of any successful software system.

This chapter presents the data structure designed in this research project for the LIME that is specifically for large scale power system studies. The specific considerations are data management problems for very large scale power system network one-line diagrams and the network device parameters, and the required flexibility for accommodating different data structures for power system application programs available commercially and academically. Based on the analysis of the features and requirements for such a LIME, the following data tables have been designed for

- Network topology description,

- One-line diagram drawing,
- Element parameter description,
- Data relationship description.

A viewport projection matrix for managing diagrams has also been defined. With this design the data management for very large scale power system network one-line diagrams and parameters is efficiently supported, and the compatibility with various data structures for different application programs is achieved. Such compatibility is realised by allowing users to redefine their own data structures. This design is one of the important contributions of this research.

4.1 Analysis of the hardware and software environment

How to record and manage a large amount of data effectively and efficiently on PC platforms is the most difficult issue in investigating and implementing a Local Information Management Environment(LIME) for large scale power system networks. Based on the present PC hardware conditions, the available storage for an information management system contains two parts:

- internal memory with normally 4MB capacity ;
- hard disk with capacities in the range of 210MB - 540MB;

where the internal memory is the shared resource for the computer system. Software programs with relevant data will be loaded in the internal memory in the executions and be deleted once the computer is switched off, i.e. the information kept in the internal memory is temporary and dynamic. There is another limitation of the PC's internal memory. PC CPU is 80X86 range CPU whose addressing mode is composed of segment address and offset address. Because of the complexity of this addressing mode, the data buffer provided for users has been restricted in a segment space(64K) by the DOS operating system. In order to use extended memory, some computer

languages have provided some functions for breaking the '64K limitation' barrier. However when the application program requires a combination of several computer languages, the available data buffer still cannot exceed 64K. So the PCs internal memory, compared with the hard disk, has a much lower capacity as well as being an expensive resource. As a result of which the internal memory is generally not used for the storage of a large amount of information especially not for long term storage. The hard disks as the low cost and large capacity storage can provide both static and dynamic services for computer systems as follows:

- when the computer system is not in operation, the hard disk, as the static storage, is used to store all the data and executable code for a long period;
- during the operation of the computer system, the hard disk is used as the backing storage for the internal memory. The data and executable code stored in the hard disk will be partly selected to be copied to the internal memory as required by the execution or processing process. The data in the hard disk is dynamically read, written and updated by the executing program.

Because of the above characteristics of computer storage, data flow scheduling has been invoked when the amount of data to be processed is over the capacity of the internal memory. There are various data flow scheduling methods. The main objective is to sequentially load in the data, which is being required by the processing procedure, from the hard disk to the internal memory.

The basic data flow scheduling method is to load in the data file fragment by fragment to the internal memory. The file is fragmented according to the available internal memory capacity. Before loading in the next fragment, the fragment currently in the internal memory, if there has been any updating, should be written back to the hard disk then the data buffer should be erased. This method is simple and is mainly used

in scheduling data flow for file editing programs. The limitation of this method is that the user has to access the data file fragment by fragment sequentially and it is impossible to directly index access a specific part in a fragment and skip over any others.

A different method for scheduling data flow is used for data and information that can be classified and related by their characteristics. Using relational keywords and indexing techniques the specific information can be retrieved from large amounts of data in the hard disk and loaded in to the internal memory without the sequence restriction. The most successful techniques used for this kind of scheduling are the relational database techniques. Presently there is a number of relational database packages that are commercially available such as PARADOX and dBASE IV. These database systems provide a complete set of data management functions, which can be used to create database data tables, define data structures, define indexing keywords and data relations. The data then can be accessed by using Structured Query Language(SQL) associated with the database systems. The relational database system advantages on data management have provided the possibility of scheduling data flow by the index method.

In addition a further consideration for the development of an information management environment for power system networks is the compatibility of today's popular software packages to provide users with sufficient tools to deal with complex tasks. For this research project, therefore, the available database development tools have been surveyed to meet the following requirements and the PARADOX Engine was selected as a result:

- popular
- reasonable price

- capability of linking with C
- capability of linking with MS-Windows application programs
- compatible with other database systems

PARADOX is a popular database system for MS-Windows. PARADOX Engine provides a complete set of relational database functions that can be used to link PARADOX and C. Therefore the combination of PARADOX Engine, C and the application program interface of MS-Windows can provide all the engineering database functions. The relational database data tables and index files created through the PARADOX Engine are 100% compatible with some popular relational databases such as PARADOX and dBaseIV.

The above selection for the database system support in the LIME implementation provides only the possibility of using the index method for data flow scheduling. The next important issue is how to use database techniques effectively to construct a distributed information management environment for power system analysis and studies. The main considerations include

- low data redundancy,
- effective data indexing,
- effective data scheduling,
- lowest data redundancy in communication,
- effective data relationships,
- flexible data structure,
- effectively record one-line network diagram information.

In summary the basic requirements are the system efficiency and data redundancy reduction. How to raise system efficiency and reduce data redundancy has been of interest to the research area of database techniques. The two goals actually are

interrelated and sometimes even contradictory. In the use of relational databases an extreme reduction of data redundancy may also cause the reduction of system efficiency; to achieve system efficiency will normally require corresponding data redundancy. Therefore the common way in engineering applications is to allow a reasonable data redundancy for the necessary system efficiency, although theoretically the redundancy can be reduced to the lowest level according to the normal form analysis. This is also the route adopted by this research, where the interest is to use popular database techniques to support the implementation of a distributed information management environment rather than to evaluate or develop the database techniques.

4.2 Connection keywords

A power system network is composed of a number of device elements that can be categorised as busbars, loads, transformers, cables(lines) and generators. These devices are connected to each other. To describe their relationships with one-line diagrams, connection keywords have been commonly used by most of power system application programs, where

- for each element connected with two busbars (such as cables and transformers): record the element parameters with the description keywords of the two related busbars, e.g. FromBusbarKey, ToBusbarKey, ElementName, ElementParameter ;
- for each element connected with only one busbars (such as loads and generators): record the element parameters with the description keyword of the related busbar, e.g. FromBusbarKey (or ToBusbarKey), ElementName, ElementParameter.

There are two ways to record FromBusbarKey/ToBusbarKey:

(1) Identify busbars by numbers

Using numbers as busbar identity keys is a simple way. It is suitable for networks that are not very large such as small scale network model for academic research. For large scale power system networks this identification method may cause a poor file readability.

(2) Identify busbars by names i.e. ASCII strings

Using names as busbar identity keys is a more suitable way for professional power system networks. In most cases in such a network each element is named with a word related to its geographical location. Thus when a fault happens to a certain network element the recovery working area can easily be identified.

Both of the above ways on identifying busbars have been adopted in the LIME implementation in this research work to suit various power system network applications.

4.3 Data tables for power system description

In the above method of describing element connections, the connection keywords are recorded together with element parameters. However this has to be reconsidered for a local information management environment in a distributed system, where the updating of power system element parameters and network topology descriptions may be required separately. This is because, firstly, in a distributed system different local environments should be able to provide individual services for different applications, so different element parameters may be required for the same power system network, i.e. the same network topology. Secondly, the network topology and one-line diagram are regularly updated by the system engineer in the main database of the host computer and then sent to each local environment database. In this process element parameters need not to be involved since they should be updated in the local

environments. Therefore for reducing data redundancy in the communication it is necessary to alter the above form used for element data records into two independent forms, one for element parameters, one for element connections, i.e. network topology. Besides, although the network one-line diagram data is sent together with the information about element connection descriptions, only the latter is required with the element parameters for the calculation programs in the local environments. This suggests further independent relationships between element connections data record and network diagram data record, which should be considered for efficient data scheduling. Thus the following three data tables have been designed for the implementation of the LIME: Network topology description data table, Drawing diagram data table and Element parameter data tables. Fig.4.1 illustrates these data categories and their roles in system operation.

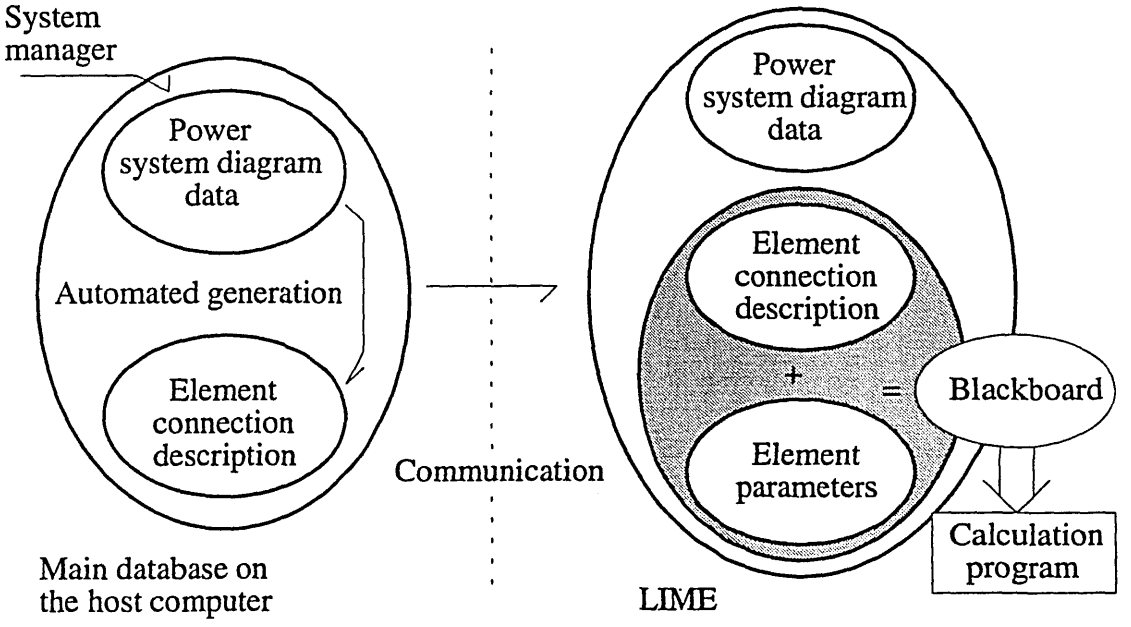


Fig.4.1 The roles of data categories in system operation

4.4 Design of the data structure for the network topology table

4.4.1 Problems in existing network topology description method

To design data structure for the Network topology data table, present method for network topology description has been investigated. Consider a simple example to explain the problems in that kind of method. Suppose there is a simple power system network as shown in Fig. 4.2, where BusA, BusB and BusC stand for Busbar A, Busbar B and Busbar C respectively; T_1, C_1, L_1 and G_1 stand for Transformer 1, Cable 1, Load 1 and Generator 1 respectively.

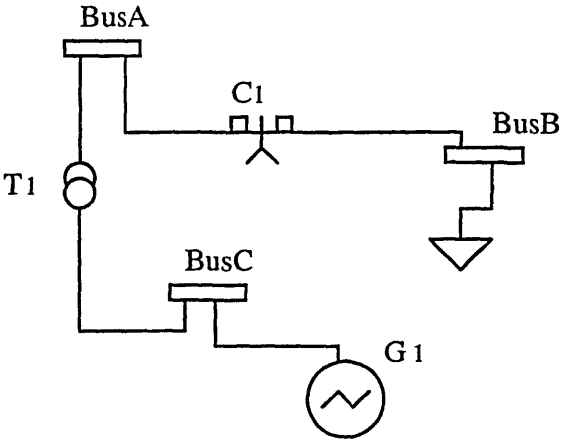


Fig. 4.2 An example of power system networks

In the present method the network topology is described by table 4.1

FromBus	ToBus	Element Name	Element Class
BusA	BusB	C_1	C (Cable)
BusC	BusA	T_1	T (Transformer)
BusB		L_1	L (Load)
	BusC	G_1	G (Generator)

Table 4.1 Present method for network topology description

Here any element of classes C, T, L and G is described by three data fields:

- FromBus
- ToBus
- Element Name

The information about Busbars contains two fields, FromBus and ToBus. This requires two index files with two corresponding search subroutines for busbars. Suppose the two index files are IndexFile 1 for FromBus and IndexFile 2 for ToBus , two subroutines are Subroutine 1 for FromBus and Subroutine 2 for ToBus; the roles of these index files and subroutines can be explained by the following 'Rename BusA with BusD' example; the operation procedure for which is given in Fig. 4.3.

From the example given in Fig.4.3 it is elucidated why two index files and two search subroutines are needed. For the proposed LIME, however, any one index file for large scale power system networks will require a large amount of storage space, and the time required by any one search subroutine will also affect system efficiency. In this respect the present network topology description method is comparatively computationally expensive and inefficient.

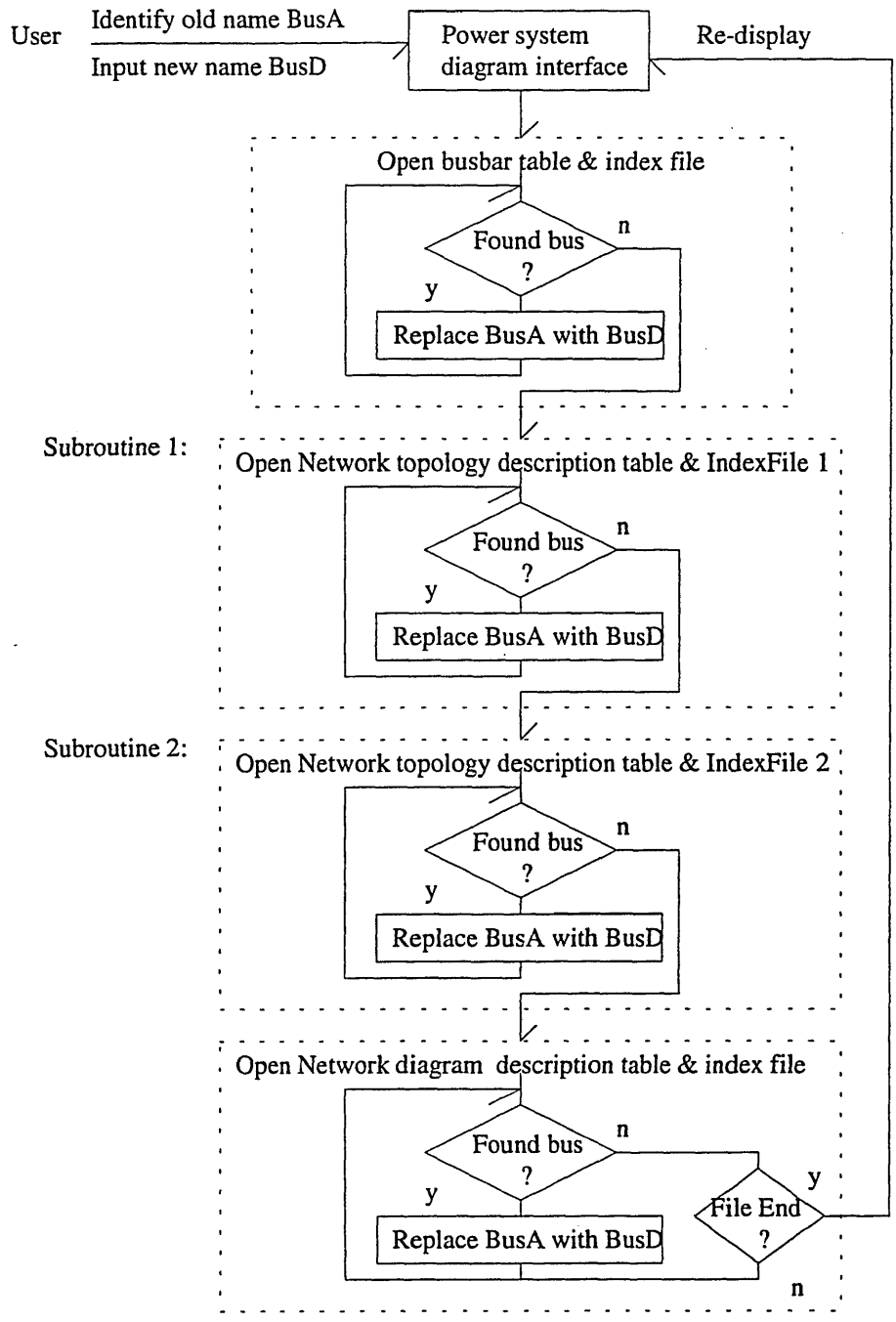


Fig. 4.3 The procedure for the 'Rename' example

4.4.2 Design of the data structure based on network element characteristics

To solve the problems appearing in the present method, the design of the data structure for the Network topology table has been based on the analysis of the characteristics of the power system network elements. The network elements can be

categorised into five device classes: Busbar (B), Cable(C), Transformer(T), Load(L) and Generator(G), and called B elements, C elements, T elements, L elements and G elements respectively. These element classifications with their description symbols and characteristics (explained later) are given in Table 4.2. These elements are connected to each other in a power system network. The connection rule is that any one B element can connect with a number of other elements, and any of C, T, L and G elements must be connected with B elements, where

a) each G element or L element just has one input or output port, so it is connected with only one B element;

b) each T element or C element has both input and output ports, and it is connected with two different B elements.

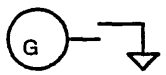
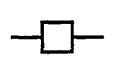
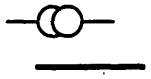
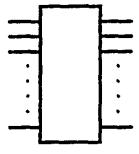
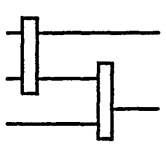
Classification	Symbol	Example	Characteristic
G L			lodger element
T C			connection element
S			lodge element

Table 4.2 The presentation of the power system elements

Based on the connection characteristic that all other elements must be connected with busbar(s), the elements can be grouped by the Busbar(s) they connect to, i.e. each of the groups contains one busbar and all the elements connected with this busbar. Such a group is called a Joint Object (JO). The power system network topology can be described by a set of JOs. Suppose that JO_i (i=1,2,...) presents a set of elements relative to Busbar B_i,

$$JO_i=\{ B_i, \{G\}_i, \{L\}_i, \{C\}_{ij}, \{T\}_{iq} \};$$

where $\{G\}_i = \{ G_{i1}, G_{i2}, ...\}$ is a set of G elements connected to B_i;

$\{L\}_i = \{ L_{i1}, L_{i2}, ...\}$ is a set of L elements connected to B_i;

$\{C\}_{ij} = \{C_{ij1}, C_{ij2}, ...\}$ is a set of C elements connected to B_i, j =1,2, ...;

$\{T\}_{iq} = \{T_{iq1}, T_{iq2}, ...\}$ is a set of T elements connected to B_i; q =1,2, ...

For any device class, if no elements of this class are connected to B_i , then the above corresponding set is empty. Since JOs are built based on B elements, B elements are defined as "lodge elements". The G elements of $\{G\}_i$ and L elements of $\{L\}_i$ are defined as "lodger elements" of B_i. The C elements of $\{C\}_{ij}$ are not only connected with $B_i \in JO_i$ but also connected with $B_j \in JO_j$, so they are defined as "connecting elements" of B_i and B_j ($i \neq j$). Similarly T elements of $\{T\}_{iq}$ are defined as "connecting elements" of B_i and B_q ($i \neq q$). The JO relations are shown in Fig 4.4.

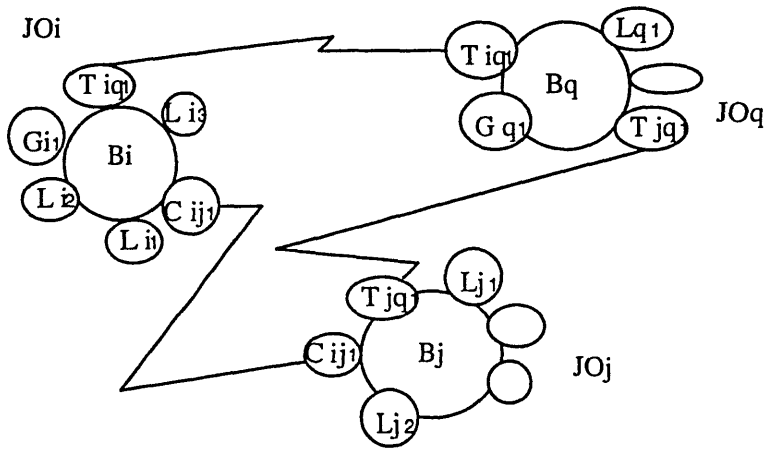


Fig .4.4 The relations of JOs

Based on the JO concept, the element connections in the example network given by Fig. 4.2 can be described as:

Record#1: BusA, C_1

Record#2: BusA, T_1

Record#3: BusB, C_1

Record#4: BusB, L_1

Record#5: BusC, T_1

Record#6: BusC, G_1

Generally, JO data structure is composed of fields of Busbar Name and Device Name. Each B element is unique in its JO so that it is used as the primary key for indexing the corresponding JO. For example, indexed by busbar B_i, all the elements connected with B_i are identified as JO_i elements. Meanwhile C elements, L elements, T elements and G elements are used as the secondary index keys for indexing their lodger elements, i.e. the connected B elements. For example, indexed by C_{ij} , B_i and B_j will be identified as the lodger elements of JO_i and JO_j. So the use of the JO data structure has provided a complete connection description with an effective indexing method in the relational database for a power system network.

The design of the data structure for network topology description is based on the JO data structure. Furthermore, device class is an important characteristic of the element and should be identified by the data structure. Also the breaker status will directly effect the element connections and should be recorded. Therefore the data structure for network topology description table is defined as shown in Table 4.3.

Busbar name	Device name	Device class	Device breaker

Table 4.3. Network topology description table

Compared with the traditional data structure discussed in section 4.4.1, the data structure developed here needs only one Busbar index field thereby one search subroutine. This means the reduction of one index file and one search subroutine, and, therefore, a saving in the system cost and an increase in system efficiency.

4.5 Design of the data structure for the diagram drawing data table

The objective of the data structure design for diagram drawing task is to use database techniques to record and manage diagrams for very large scale power system networks on PC platforms. Aided by computer systems, the usual way to draw a two-dimensional diagram is by using the Vector method. For example, supported by Microsoft C drawing functions, a straight line can be easily drawn by specifying the co-ordinates of the start point and end point; the addition of an arc is to specify the co-ordinates of the upper-left and lower-right corners of the bounding rectangle. These are illustrated in Fig. 4.5.

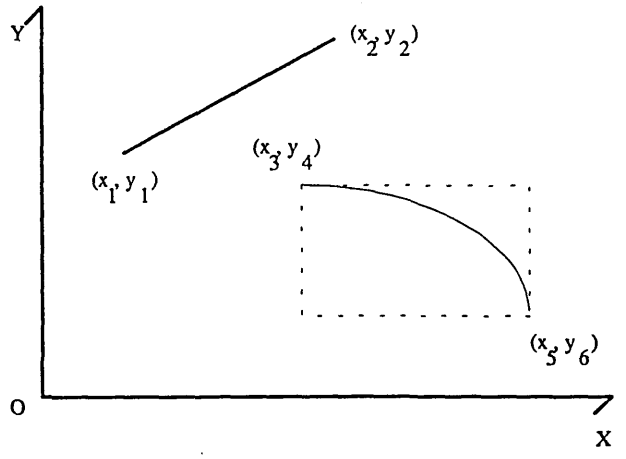


Fig.4.5 Drawing a line and an arc

Generally the Vector method requires very little data to describe and locate a diagram element in a co-ordinate system. Thus one-line diagrams for power system networks are normally drawn using this method. This is also the basic method adopted by this research for drawing power system diagrams. The problem is how to use this method and database techniques to support diagram management for very large scale power system networks.

The data structure design was started with the analysis of the basic way to describe the location of an diagram element and its input/output connection lines in a co-ordinate system. Firstly the location of the centre point of a diagram element is defined as the location of the element. Suppose that

(x_e, y_e) is the location co-ordinate of element E,

(x_I, y_I) is the input connection point of element E,

$\left(x_{I_1}, y_{I_1}\right)$ is the first turning point on the input connection line of element E,

$\left(x_{I_2}, y_{I_2}\right)$ is the second turning point on the input connection line of element E,

(x_O, y_O) is the output connection point of element E,

$\left(x_{O_1}, y_{O_1}\right)$ is the first turning point on the output connection line of element E,

$\left(x_{O_2}, y_{O_2}\right)$ is the second turning point on the output connection line of element E;

then the diagram of element E and its connection lines can be drawn as shown in Fig.4.6

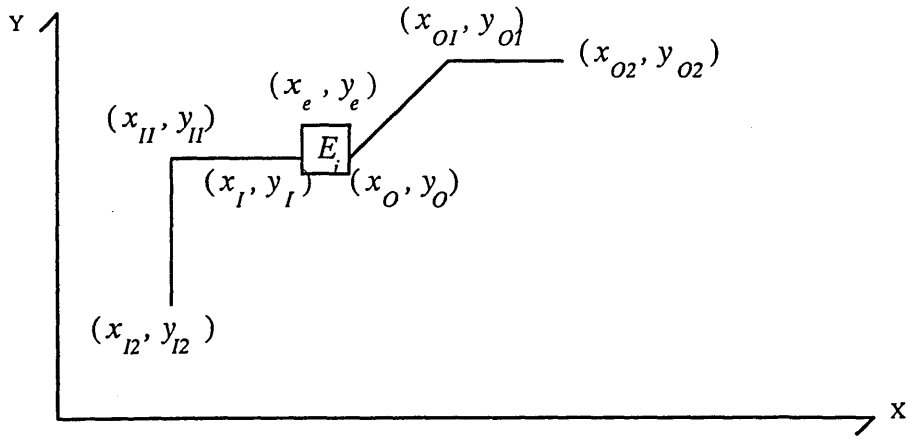


Fig.4.6 Element E with its connection lines

Theoretically all other elements of the power system network can be drawn in the same way, i.e. a complete one-line network diagram. However as the computer screen has a limited size, it will not be possible to fit large scale one-line diagrams to one screen. A way to overcome this is to use **drawing board** and **viewport** concepts, defined as follows, to keep the network diagram drawn and viewed part by part on a continuous basis.

***Definition* 1** A drawing board is a logical plane with a co-ordinate system .

***Definition* 2** A viewport is a rectangular part of the drawing board that can be viewed through the display window on the computer screen.

Suppose (x_0, y_0) is the origin of the co-ordinate system on the drawing board,

x_1 is the x-co-ordinate of the upper-left corner of the viewport,

y_1 is the y-co-ordinate of the upper-left corner of the viewport,

x_2 is the x-co-ordinate of the lower-right corner of the viewport,

y_2 is the y-co-ordinate of the lower-right corner of the viewport,

(u_0, v_0) is the origin of the co-ordinate system on the viewport, where $u_0=x_1$, $v_0=y_2$,

the co-ordinates of element E and its connection lines on the drawing board are the same as that given above;

then the position of the viewport on the drawing board can be described by x_1, x_2, y_1, y_2;

and the co-ordinates of the element E under the co-ordinate system on the viewport is:

$$u_e = (x_e - x_1)$$

$$v_e = (y_e - y_1)$$

Fig. 4.7 illustrates the relations between the drawing board and the viewport.

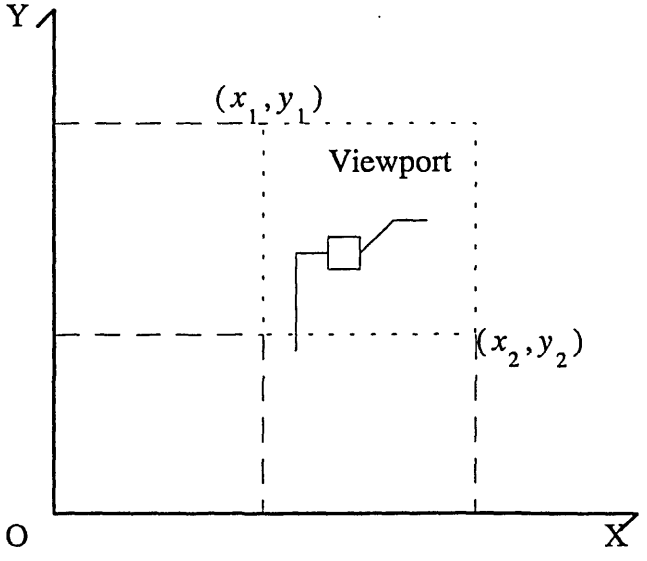


Fig. 4.7 Drawing board and Viewport

Based on the above definitions and assumptions a diagram element of a power system network can be described in the following structure:

i, name, class, x_e, y_e, x_I, y_I ,x_{I_1} , y_{I_1} ,x_{I_2} , y_{I_2} , x_O, y_O, x_{O_1} , y_{O_1} ,x_{O_2} , y_{O_2};

where i is the record number, i =1,2,...; 'name' is the element name; 'class' is the element class including Busbar (B), Cable (C), Load(L), Transformer (T) and Generator(G).

This description has fixed each diagram element in a unique location, therefore the one-line network diagram itself, on the drawing board. Note there is no limitations to the number of elements. For a very large scale power system, the area covered by the network diagram may be much much larger than the viewport. In this case it is supposed that the diagram can be viewed by moving the viewport all over the drawing board. In fact 'moving viewport to view diagram' is the equivalent of the action to identify and display the diagram elements that are currently in the viewport rectangle.

Suppose that there are m diagram element records in the hard disk, the capacity of the memory buffer is n records, and $m >> n$; then the task is to select the elements whose co-ordinates are in the current viewport rectangle area and to display these elements, i.e. that part of the one-line network diagram. This means, using the previous assumptions, reading n_1 $E(x_e, y_e)$ records from the m records in the hard disk and writing to the buffer for display, where $n_1 \leq n$, $x_1 \leq x_e \leq x_2$, and $y_2 \leq y_e \leq y_1$. Because there is no relation between the record number i (i =1,2,...) with the co-ordinates of element E_i , the selection has to be done by searching all the m records in the hard disk, i.e. searching m times to load all the n_1 satisfied records to the buffer. The search program flowchart is shown in Fig.4.8.

The Program Flowchart:

BP: the buffer pointer

FP: the data file pointer

x_1 : the x-co-ordinate of the upper-left corner of the viewport

y_1 : the y-co-ordinate of the upper-left corner of the viewport

x_2 : the x-co-ordinate of the lower-right corner of the viewport

y_2 : the y-co-ordinate of the lower-right corner of the viewport

x_e : the x-co-ordinate of the element E

y_e : the y-co-ordinate of the element E

BMax: the buffer maximum record number = n

FMax: the data file maximum record number = m

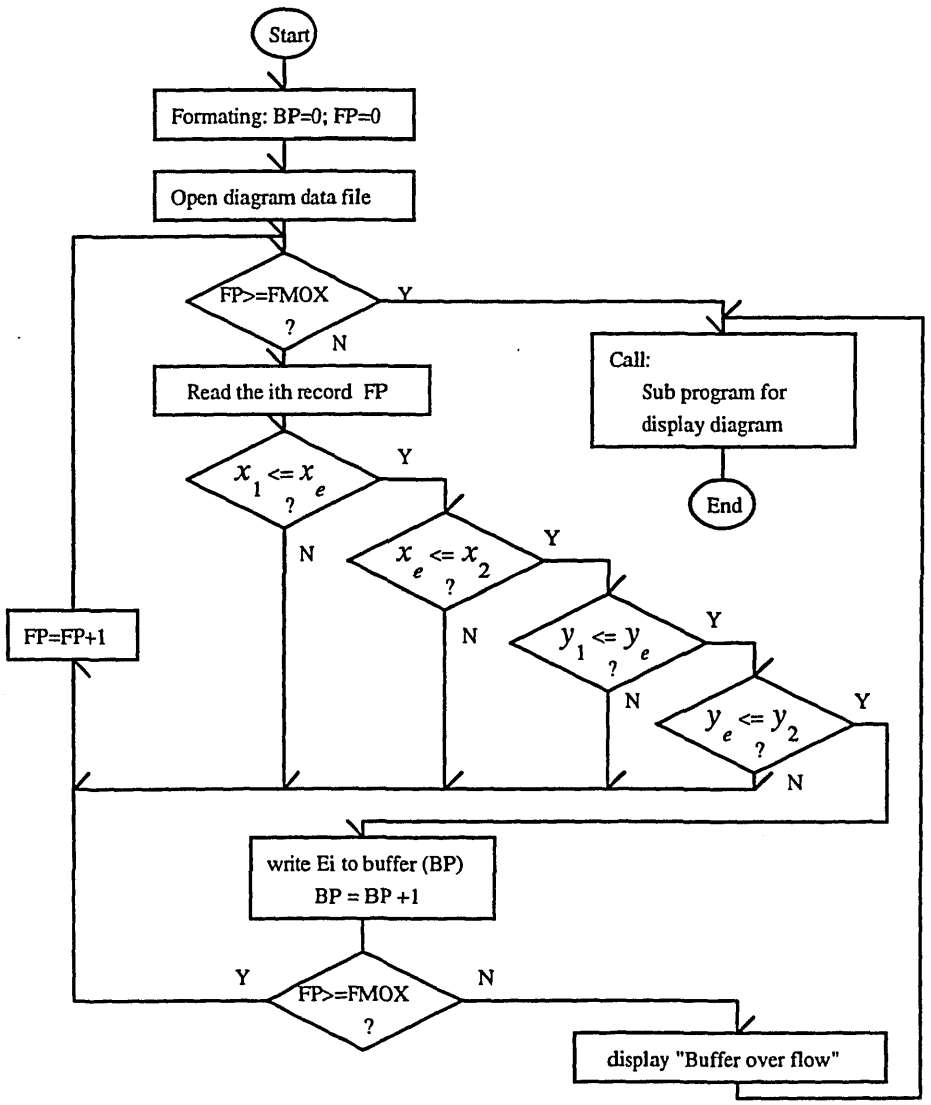


Fig.4.8 The search program flowchart

The problem here is that the search program has to search the hard disk m times for each time the viewport is moved. Thus, when $m >> n$, the diagram view operation could be too slow to be applicable. There are two major reasons creating this problem.

First, there is no identify key to specify the co-ordinate interval for the element. Therefore for selecting one element the searching program has to make the following four co-ordinate comparisons:

$x_1 \leq x_e$,

$x_e \leq x_2$,

$y_2 \leq y_e$, and

$y_e \leq y_1$.

Second, the sequence of the element records is not naturally related to its location area on the drawing board. When a rectangle area on the drawing board has been identified as the current viewport, it cannot index the record addresses in the data file for the elements within the rectangle area. There are, again, two explanations for this irregularity between the element record sequence in the data file and the element locations on the drawing board.

(1) Element drawing sequence

Suppose that the drawing board is divided into a number of rectangles and *A*, *B*, *C*, *D* are four of these rectangles and located as shown in Fig. 4.9.

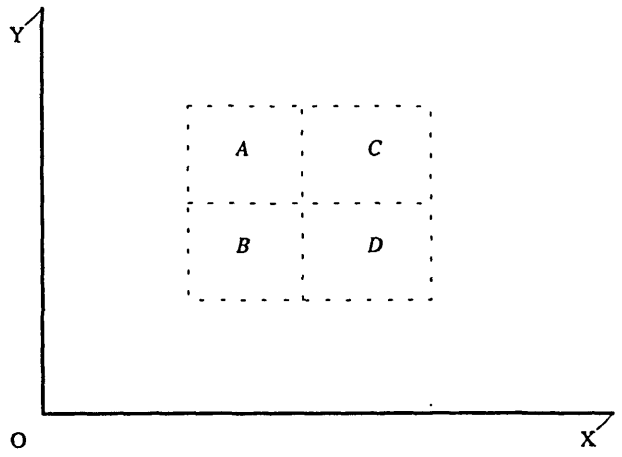


Fig.4.9 Rectangles *A*, *B*, *C*, *D* on the drawing board

For example, if the sequence of the user drawing element E_i (i=1,2,...m) is as shown in Fig.4.10, then the records of the elements in area A could be distributed in any position in the data file. So to find all the elements that locate the area A requires searching from the beginning to the end of the data file.

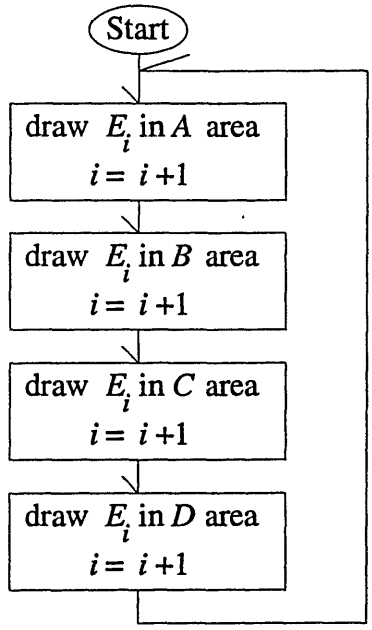


Fig.4.10 An example of the user drawing sequence

(2) The difference between element record and element location description

As has been seen, the distribution of diagram elements is described by a two-dimensional co-ordinate system, but the way by which the elements are recorded in the computer hard disk basically is a one-dimension data table. Therefore any specific diagram area cannot index the addresses in the data file for the elements of this area.

It seems better to build a two-dimensional matrix to record elements data. Then the record of any element $E(x_e, y_e)$ on the drawing board can be indexed by its mapping

element $m_{x_e y_e}$ in the matrix. However there are two problems. Firstly, in the drawing board, not every point has to be filled by an diagram element, so the projection of the diagram element to the matrix is a sparse matrix, i.e. a matrix with most elements having 0 values. But for the computer language to write a two-dimensional structure data file, it requires first to build a matrix description and assign corresponding space before recording the data. A sparse matrix will therefore occupy the same space with a matrix fully filled with non-0 values. This is obviously a considerable waste of the hard disk storage space. Secondly, the data records managed by relational database systems is in a one-dimensional structure, thus the two-dimensional matrix structure cannot be well supported by the database techniques.

It could be concluded from the above analysis that, in order to solve the display speed problem for viewing large diagrams through the viewport, the following is required:

- *set an unique location index key for each element record,*
- project the diagram elements' two-dimensional co-ordinates to a one-dimensional data table.

Therefore a viewport projection matrix has been defined which will dynamically produce the location IndexKey for each diagram element record.

Since the movement of the viewport is actually a combination of move-down/up and move-left/right, a unit moving step can be used to measure horizontal and vertical moving distances and therefore identify the current location of the viewport.

Definition 3 An *unit moving step* is a fixed length for measuring the viewport horizontal and vertical moving distance. It is denoted by $|\Delta u| \neq 0$, and

$\Delta u > 0$, for move-up and move-right;

$\Delta u < 0$, for move-down and move-left.

Definition 4 A *co-ordinate string* is an n-character string to present an m-digit x-co-ordinate or y-co-ordinate by using always the last right m digits of the n characters, where $m \leq n$ and the characters from the first left to the $(n-m)$th digits of the string are zero.

For example, using a 5-character co-ordinate string, an x-co-ordinate of 24 will be presented as 00024, and a y-co-ordinate of -6 is 000-6.

Definition 5 A *co-ordinate string combination*, denoted by $\oplus$, is an ordered appending of two co-ordinate strings. The character digit of the combination is the sum of the two strings digits.

For example, the combination of the above first and the second string is

00024⊕000-6=00024000-6.

The combination of the above second and first string is

000-6⊕00024=000-600024.

Definition 6 The sum of an n-character co-ordinate string and a number is the n-character co-ordinate string of the sum of the co-ordinate and the number.

For example, 00024+8=00032 and 000-6+10=00004.

Definition 7 Suppose α is an n-character co-ordinate string of the x-co-ordinate of the upper-left corner of the viewport, β is an n-character co-ordinate string of the y-co-ordinate of the upper-left corner of the viewport, $|\Delta u|\neq 0$ is the unit moving step, the area covered by the viewport is $(m|\Delta u|)^2$; then a *viewport projection matrix VM* is defined as follows:

$$VM=\{v_{ij}\}=\begin{bmatrix} 0 & \alpha & \alpha+|\Delta u| & \alpha+2|\Delta u| & \ldots & \alpha+m|\Delta u| \\ \beta & 1 & 2 & 3 & \ldots & m+1 \\ \beta+|\Delta u| & m+2 & m+3 & m+4 & \ldots & 2(m+1) \\ \beta+2|\Delta u| & 2m+3 & 2m+4 & 2m+5 & \ldots & 3(m+1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta+m|\Delta u| & mm+m+1 & mm+m+2 & mm+m+3 & \ldots & (m+1)^2 \end{bmatrix}$$

For example, for α=00001, β=00010, $|\Delta u|$=1, m=4, the VM matrix become:

$$VM=\{v_{ij}\}=\begin{bmatrix} 0 & 00001 & 00002 & 00003 & 00004 & 00005 \\ 00011 & 1 & 2 & 3 & 4 & 5 \\ 00012 & 6 & 7 & 8 & 9 & 10 \\ 00013 & 11 & 12 & 13 & 14 & 15 \\ 00014 & 16 & 17 & 18 & 19 & 20 \\ 00015 & 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

In the $VM=\{v_{ij}\}$, v_{ij} (i,j =2,3,..., m) has a duplex characteristic: its value is a one-dimension number and its position in the matrix is described by a two-dimensional subscript. Using the viewport projection matrix, therefore, an index from a two-dimension location co-ordinate to a one-dimension data record number can easily be constructed.

If v_{ij} (i,j =2,3,..., m) is used to describe m^2 locations in the viewport and its location (vl_{ij}) on the drawing board is described by the co-ordinate string combination $v_{1j} \oplus v_{i1}$ = (α+(j-2)$|\Delta u|$)$\oplus$(β+(i-2)$|\Delta u|$). Then the viewport can be covered by a number of small squares with the centres vl_{ij} and side length $|\Delta u|\neq 0$. Meanwhile the value of v_{ij} (i,j =2,3,..., m) is used as the number of the buffer for an element record.

Now if the user draws an element whose centre point is in any one of the vl_{ij} (i,j =2,3,..., m) squares, then the element location can be specified by $v_{1j} \oplus v_{i1}$. Here v_{1j}

$\oplus\ v_{i1}$ is used as a location IndexKey (I_{ij}) and it is written with the element record into the buffer pointed by the value of v_{ij} . Finally all these records in the buffers will be written to the element data file in the hard disk. In the above example, if the user draws an element in the square centred vl_{43}=(2,13), then the location IndexKey is produced by $v_{13} \oplus v_{41}$= 0000200013, which will be written with the element record to the 12th buffer since the value of v_{43} is 12.

A similar method is used to view an existing diagram through the viewport. Each of the m^2 locations will be checked sequentially, if there is an element in any one of the vl_{ij} (i,j =2,3,..., m) squares, then IndexKey (I_{ij}) can be identified by $v_{1j} \oplus v_{i1}$. Indexed by I_{ij} the corresponding element record in the data file kept in the hard disk can be identified and then written to the buffer identified by the value of v_{ij} . Later when all other elements in the viewport have been written to their corresponding buffers, the diagram in the viewport will be displayed.

When the viewport has been moved to view an other part of the diagram, using $|\Delta u| \neq 0$ to give a measure of the movement, the new co-ordinates of the upper-left corner of the viewport can be specified. Then the corresponding $VM = \{v_{ij}\}$ can be produced. For example, if the viewport has moved one $|\Delta u|$ to the right, then the co-ordinate string of the x-co-ordinate of the upper-left corner of the viewport $\alpha' = \alpha + |\Delta u|$, the co-ordinate string of the y-co-ordinate of the upper-left corner of the viewport β is not changed. Then the $VM = \{v_{ij}\}$ for the current viewport can be produced.

The design and use of the viewport projection matrix has provided a solution to the two problems discussed earlier in drawing and viewing large one-line network diagrams. Supported by the viewport matrix, the data structure for the drawing diagram table was then designed as shown in Table 4.4.

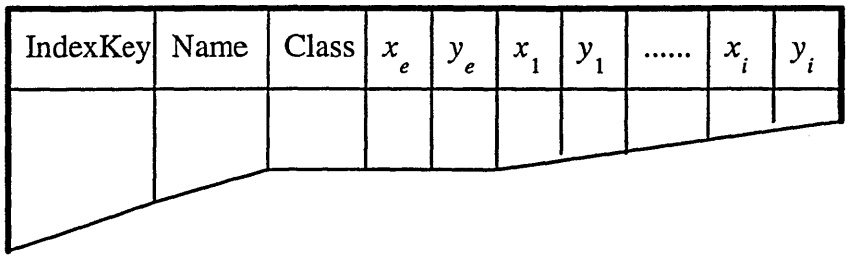


Table 4.4 Drawing diagram data table

4.6 Design of the data structure for the element parameter table

Here the key issue is how to deal with various ASCII input files in the development of a LIME for power systems. At present a number of power system application programs have been widely used for either academic or professional work. The variety and complexity of these application programs require corresponding input ASCII files with specific data structures. The result reports will also be produced in the form of ASCII files. The input ASCII files for one specific application program normally cannot be used for other programs. For work involving the use of more than one application program, the user has to make a considerable effort to re-create ASCII files in the specified structure. Therefore the open system concept has been suggested to support data sharing for different application programs. Approaches to the development of open systems based on some existing application programs have been investigated. For the protection of normal operation, however, such open systems do not allow users to update the existing input ASCII file structure. So for application programs requiring any other input ASCII files, the benefits of data sharing from the open system is, at least, not directly available. For improvement some approaches attempted to develop a 'full data set' for the open systems allowing users to select any specific data needed. The problem is the life cycle of the full data set. As new

application programs are always being investigated and developed there will not be any long lasting 'full data set'.

In summary it could be concluded that an awareness of the following is required in the development of the LIME:

- do not restrict input ASCII files to some fixed data structures;
- do not attempt to provide a *full* data set .

Therefore the possible way of meeting the requirement of various ASCII file structures for different application programs is to allow users to define their own data structures where necessary.

'To allow users to define' means that first of all the data structure cannot be designed for only a fixed form, e.g. must be defined with any fields, field sequence, field length required by users. Then, an element parameter set should cover a wider range, i.e. the parameters can be any data sets shown in Fig 4.11[GONE 84].

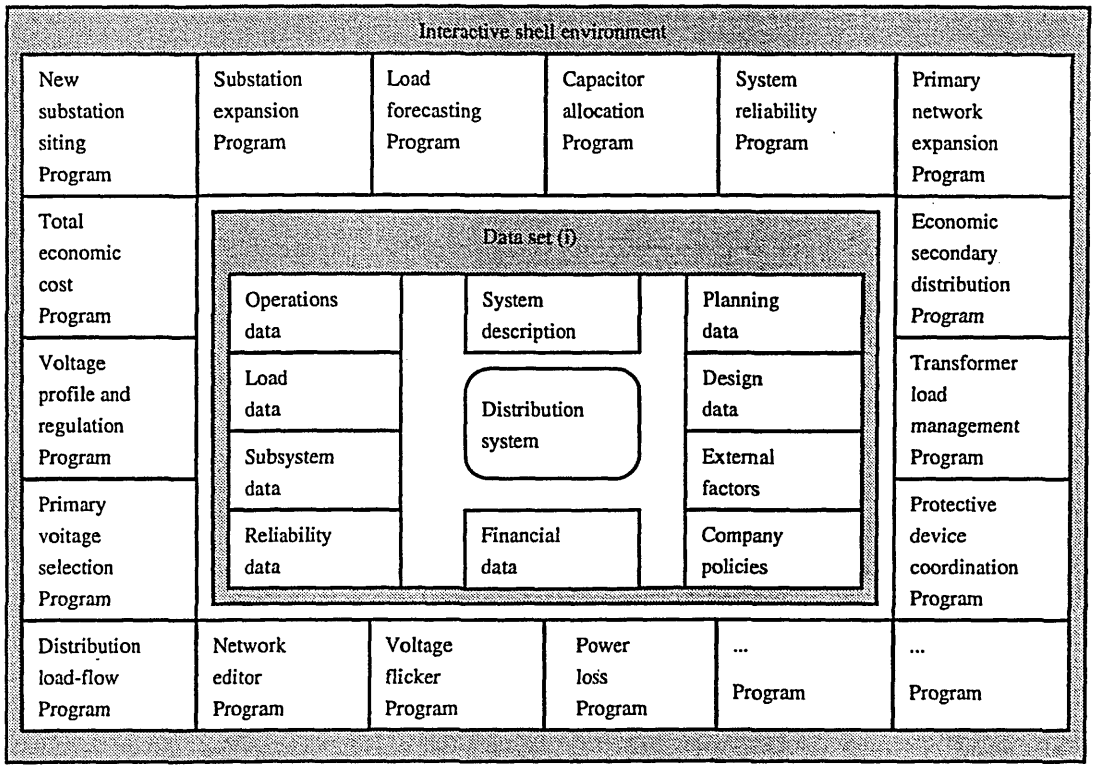


Fig 4.11 Data set table

A further technical issue is how to build indexes between the parameters defined by the user and other data in the database and therefore to keep all the data completely indexed. For this purpose five parameter table have been defined and used. This has been based on the data structure designed for Network topology description (section 4.4.2) and Diagram drawing (section 4.5). As defined previously, the fields in the Network topology description table are Busbar Name, Device Name, Device Class and Device Breaker; those of the Diagram drawing table are IndexKey, Device Name, Device Class and some corresponding co-ordinates. The fields related to the element name in these two tables are Busbar Name and Device Name. So if a Name field is also established for the element parameter table, then using 'Name' as the index key, a complete indexing for these three data tables can be achieved. Therefore the general data structure have been designed for the parameter tables as

Parameter{

char Name

char Parameter(i)

}

i=0, 1, 2,..., 79.

Considering that in the power system information management specific categories of elements may be required individually, five categorised element parameter tables have been designed as follows:

(1) Busbar table

Name	P(1)	P(2)	.....	P(i)

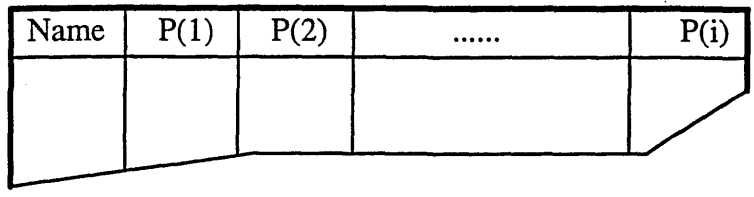
(2) Cable(Line) table

Name	P(1)	P(2)	.....	P(i)

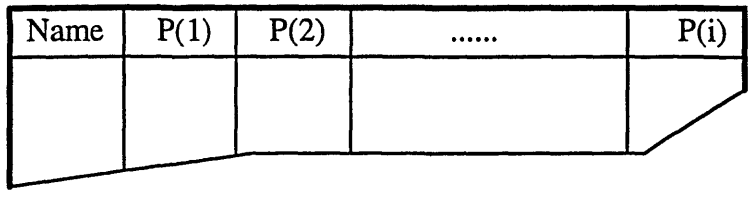
(3) Transformer table

Name	P(1)	P(2)	.....	P(i)

(4) Load table



(5) Generator table



In the above five tables the Name, used as the index key, is the fixed field and cannot be changed by end-users. Users are allowed to redefine all the other fields according to their specific requirements, which includes the update on the name, length and the position of the field. A LIME based on this design can suit most input ASCII file structures required for various power system application programs.

4.7 The Relationships between database tables

Seven tables have been designed in section 4.4-4.6. Using these tables the user can describe the network diagram, element connections and parameters for the power system. For some work such as system simulation, users may need to update some part of the power system topology, parameters or data structures, then compare the results before and after updating to optimise the system. So the LIME may be required to manage information for more than one power system project. If the user wants to create n power system projects ($n \in \{2,3,...\}$), then there will be $7 \times n$ data tables

involved. Thus the user has to remember the names of all these tables and which tables belong to which projects. The more projects are involved the more difficult the management is. Therefore a project manager has been developed by this research to help users to manage these data tables in an object oriented operation style.

Since a power system can be described by the above 7 tables, a Power System Project is defined as a set of data tables which contain:

- Network topology description table,
- Diagram drawing data table,
- Busbar table,
- Cable table,
- Transformer table,
- Load table,
- Generator table.

To describe the Power System Project operation status an Active Key has also been designed, and hence the data structure for the project manager table can be designed as shown in Table 4.5.

ProjectName	ActivityKey	TableName(1)	.....	TableName(i)

Table 4.5 Project manager data table

In this table the field 'Active' has two values: 0 or 1. If the 'Active' value is 0, then this Power System Project is in a 'sleeping' status. If the 'Active' value is 1, then this Power System Project is the currently active project, called Current Project. The Power System Project active status can be easily changed through the Project management

program as users require. In system operation, only the Current Project will be processed as the currently active object. The processing status of the seven data tables could be: open, close, read, write, update, search, or connect each other. All these data tables of the Current Project are provided by the Project manager module. Then the user need not have to manage data tables.

There are also seven Names in the Project manager table. These Names are registered to the table automatically when the user creates a new Power System Project. These Names will be used for activating any one of the Power System Projects or as the source data for the processing of relevant programs. When a Power System Project is created and registered in the Project manager table, its data relationships are completely described as illustrated in Fig. 4.12.

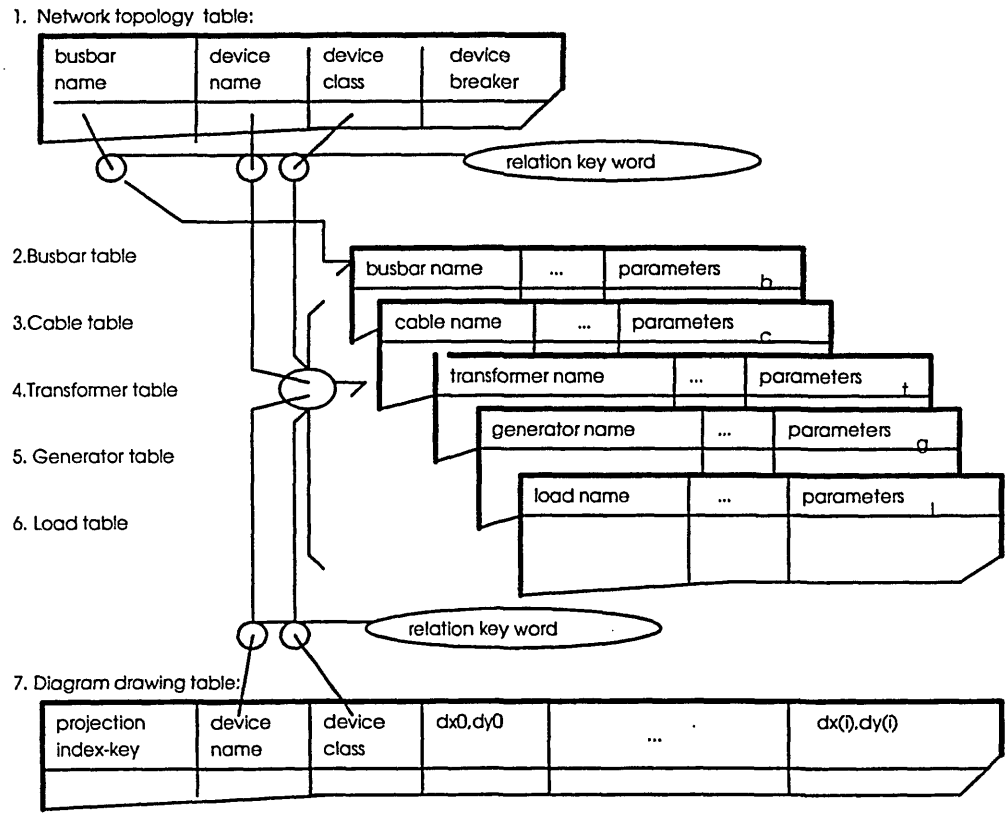


Fig .4.12 The data table relations

4.8 Summary

In this chapter, based on the analysis of the limitations of current PC hardware and software resources, the use of database techniques has been identified as the key point to enable the local PC platforms to manage one-line network diagrams and element parameters for very large scale power systems. Then considering the nature of the power system information, the research work has concentrated on the design of the data structures for constructing a relational database for the LIME. The database data tables designed have been used to describe and record:

- power system network topology,
- one-line network diagram,
- power system network element parameters,
- data table relationships.

This work has provided the data storage and index method and therefore forms the basis for information processing in system operation. This is also the basis of providing the LIME with the capability of managing information for very large scale power systems.

CHAPTER 5

POWER SYSTEM NETWORKS ONE LINE DIAGRAM DRAWING METHOD DESIGN

The research work presented in this chapter details the design of the data flow scheduling method for drawing power system network diagrams for the LIME. This data flow scheduling method design has been based on the database table data structure discussed in Chapter 4. It enables the LIME to manage very large scale power system network parameters and diagrams on PC platforms. This is one of the original contributions of this research. In this chapter the design work has been discussed in relation to both automated and CAD styled power system network diagram generation. For the automated diagram generation, the discussion focuses on a recursive method for managing the use of buffers in order to support very large scale network diagram generation. For the CAD styled diagram generation, the discussion concentrates on graphical information partitioning techniques. This includes the concepts of Drawing Boards and Projection Matrixes, and methods of scheduling Drawing Boards and drawing data by using Projection Matrices. Considering requirements specified previously, the CAD style method has been used in the implementation of the LIME. The method of automated diagram generation has been implemented in a separate prototype for a comparison. Both methods have been demonstrated as being successful in the support of diagram and parameter management for very large scale power system networks.

5.1 Automated network diagram generation

As mentioned in Chapter 2, the operation procedure for Automated Network Diagram Generation (ANDG) first of all requires the user to input the ASCII file containing the description of the network topology. The ANDG program, according to the topology description, will then assign a diagram symbol with its drawing location to each element of the power system network. The method used for network topology description and the structure of database data tables have been discussed in Chapter 4

(section 4.4). Also in section 4.5 the diagram drawing data table structure has been designed to be able to support the drawing of very large scale power system networks. Based on these data structure designs, the next step is to construct a data processing algorithm for the ANDG program to generate the network diagram with a satisfactory readability.

Computer aided diagram drawing tools use specific algorithms in generating diagrams. For example, the algorithm of Power Tool software, developed by SKM System Analysis Inc, is to distribute elements and spread the network from 'Source' (generator) to 'Load'. Actually even using the same computer aided tool, individual operators would draw a power system network one-line diagram using different schematic layout for locating network elements. This is mainly because of the operator's personal operational experience and the preference for an acceptable placement of the power system elements in the diagram. These differences do not affect the nature of the power system network since the one-line network diagram is not a physical/ geographical structure diagram but a topological graphic representation of network elements and their interconnection.

Essentially, regardless the geographical distribution of the power system network an automated diagram generation program should indicate and display:

- topological connection relations
- electricity transmission direction
- transmission position sequence
- fan-out status of loads

Therefore the placement of elements in a power system network one-line diagram should be based on the following principles:

(1) choosing the power station (generation point) as the start point

(2) for each substation, the elements in its left column are either the lodger elements of its parent substations or the connection elements connecting it with its parent substations; the elements in its right column are either its lodger elements or the connection elements connecting it with its child substations.

So when the power system elements are distributed on the integer-co-ordinate locations on the drawing board. e.g.

$$x = 1,2,..., n$$
$$y = 1,2,..., m$$

the lodge elements, i.e. 'B elements', can be placed on the even columns

$$x = 2,4,6,8,...$$
$$y = 1,2,..., m$$

the lodger elements (L or G elements) and connection elements (C or T elements) can be placed on the odd columns

$$x = 1,3,5,...$$
$$y = 1,2,...m$$

An illustration of this distribution is given in Fig. 5.1.

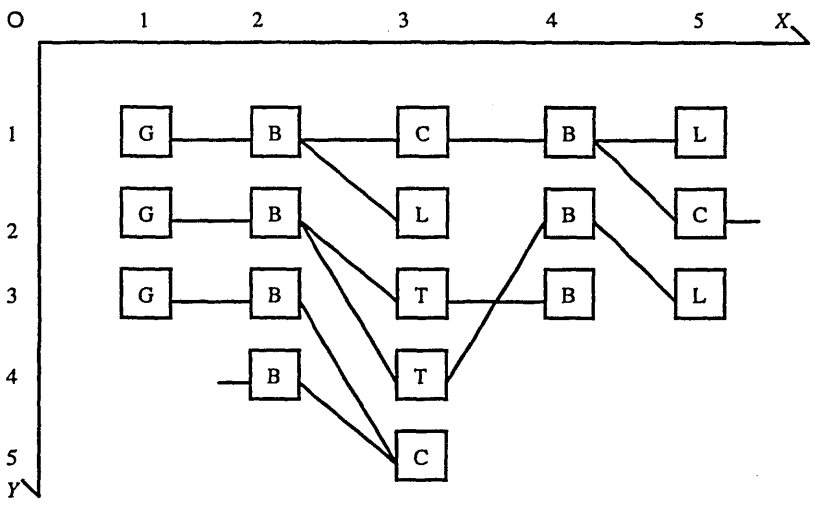


Fig. 5.1 Element distribution example

Since the elements are placed according to x =1,2,3,..., n on the drawing board, the diagram generation can be operated column by column from left to right to allow the production of large power system diagrams within a limited computer memory. Thus three buffers are used for the automated diagram generating program. These are :

- drawing board buffer,
- current JO buffer,
- scanning diagram description file buffer.

The drawing board buffer is composed of the array of two dimensions

$A=\{a_{ij}\}$ $i=1,2,...,n,$ $j=1,2,...,m.$

Each element in $\{a_{ij}\}$ is for storing the data in the data fields of the Drawing diagram table mentioned in Chapter 4(section 4.5), i.e. the following information:

a) the co-ordinate of the power system network element (x_i, y_j) on the drawing board;

b) the name of the element;

c) the element classification: B, L, C, T, G; and

d) the element record address.

The drawing board buffer can contain three columns of elements. The element placement is invoked from the top to the bottom and from the left to the right. When the current three columns in the buffer have been placed, the information indicated by above item (a) is written to the diagram description file according to item (d). Then the program changes the elements of $\{a_{ij}\}$ in the drawing board buffer as:

rewrite $a_{1j}(x_1, y_j)$ as $a_{3j}(x_3, y_j)$,
$a_{2j}(x_2, y_j)$ as $a_{3j}(x_3+1, y_j)$,
$a_{3j}(x_3, y_j)$ as $a_{3j}(x_3+2, y_j)$,
a_{1j} (name) as a_{3j} (name),
a_{1j} (key) as a_{3j} (key);

where (x_i, y_i) is the corresponding co-ordinate of a_{ij}.

Repeating this operation recursively, all the power system elements can be placed with their relationships on the drawing board.

The current JO buffer is used to load the qualified JOs specified. These JO will be decomposed, the B elements being placed on the a_{2j} column, and the L, T, G, C elements will be placed on the a_{1j} or a_{3j} column.

The scanning buffer is used for the ANDG program to scan the diagram description file part by part to find the qualified JOs and write them to the current JO buffer.

The use of the above three buffers supports the construction of a large power system network one-line diagram to be invoked in a recursive way. The following is a simple

example to explain the operating procedure of the automatic diagram generation. Suppose that the power system network diagram is as shown in Fig 5.2

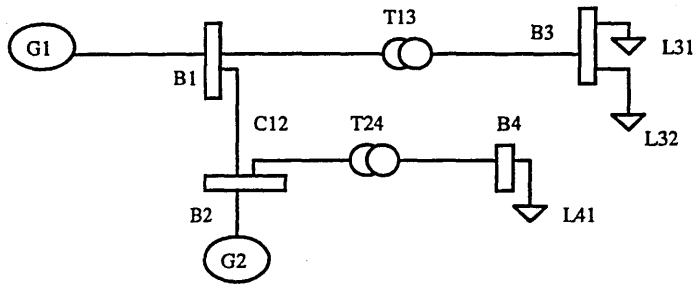


Fig.5.2 A typical power system network one line diagram

This network diagram can be described by:

$$JO_1=\{B_1, G_1, T_{13}, C_{12}\}$$

$$JO_2=\{B_2, C_{12}, G_2, T_{24}\}$$

$$JO_3=\{B_3, T_{13}, L_{31}, L_{32}\}$$

$$JO_4=\{B_4, T_{24}, L_{41}\}$$

The operating procedures for placing the power system network elements are:

(1) Placing elements according to the array in the drawing board buffer, which is shown in the table below.

j \ i	1	2	3
y \ x	1	2	3
1	G1	B1	G1
2	G2	B2	T13
3			C12
4			G2
5			C12
6			T24

The steps are:

a) Assign a_{1j} (j=1,2,..., m) co-ordinate x =1, y = j ;

b) Find the JO which contain the G elements from the power system network diagram description file, and write them to the current JO buffer;

c) Place the G element to a_{1j} (j=1,2,3,..., m);

d) Place the B elements, which are in connection with the above G elements, to a_{2j} (j=1,2,3,..., m_2);

e) Place the G,T,L,C elements, which are connected with the above S elements, to a_{3j} (j=1,2,3,..., m_3).

(2) The array is changed as shown in the table below.

j \ i	1	2	3
y \ x	1	2	3
1	G1	B1	T13
2	G2	B2	C12
3			T24
4			
5			
6			

The steps are:

a) Delete the repeat elements of a_{3j} (j=1,2,...m_3), to ensure that every element of a_{ij} (i=1,2,3; j=1,2,3,..., m) in unique;

b) Write the element co-ordinates in the drawing board to the power system network diagram description file.

(3) The array is rearranged as shown in the table below

j \ i	1	2	3
y \ x	3	4	5
1	T13	B3	T13
2	C12	B4	L31
3	T24		L32
4			T24
5			L41
6			

The steps are:

a) move the drawing board buffer to the right

b) use the T, C elements of a_{1j} (j=1,2,3,..., n_1) as an index to find the other JO in the diagram description file

c) place the B elements, which are connected with the above T, C elements, in to a_{2j} (j=1,2,3..., n_2);

d) place T, L, C elements, which are connected with the above B elements, to a_{3j} (j=1,2,3,..., n_3).

(4) The array is changed as shown in the following table.

j \ i	1	2	3
y \ x	3	4	5
1	T13	S3	L31
2	C12	S4	L32
3	T24		L41
4			
5			
6			

The steps are the same in as (2)

(5) For the third column use the operation switch key

$$K = \sum_{i=1}^{100} [\{T\} + \{C\}]$$

If K=0 then end the operation

If $K \neq 0$ then return to procedure (3)

This means that operation of the placement will be continued recursively to develop the whole power system network until K=0. At the end of the operation, every element in the power system network diagram description file is assigned to a unique co-ordinate on the drawing board. In the example network shown in Fig5.2, the element co-ordinates are generated through four recursive processes and will be used to

generate the network diagram. The illustrations for these four processes are shown in Fig.5.3-5.6 respectively, where the segmented diagrams would be the result of drawing the diagram according to the co-ordinates specified by each JO but actually not produced until K=0, i.e. the end of the network placement.

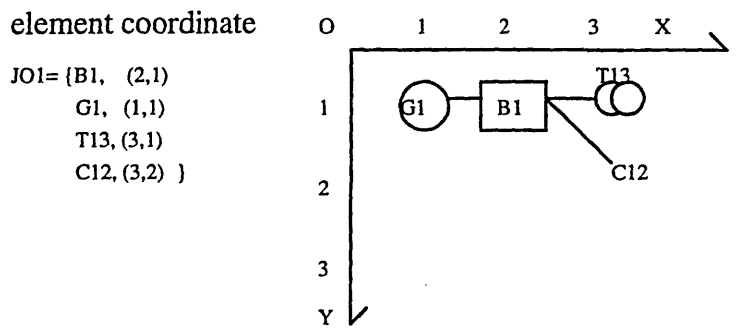


Fig 5.3 First process in diagram generation

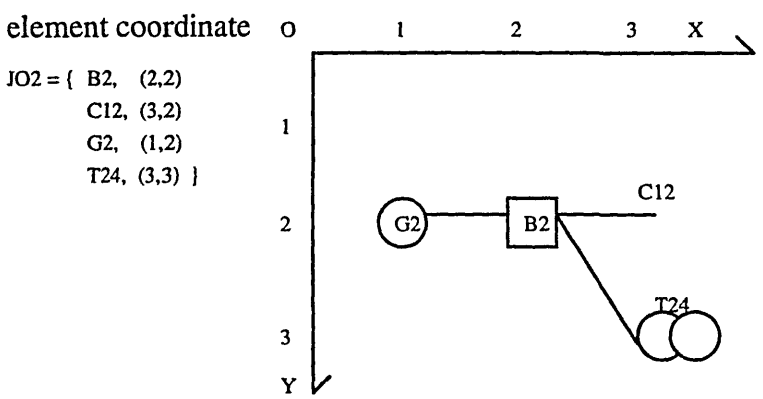


Fig 5.4 Second recursion in diagram generation

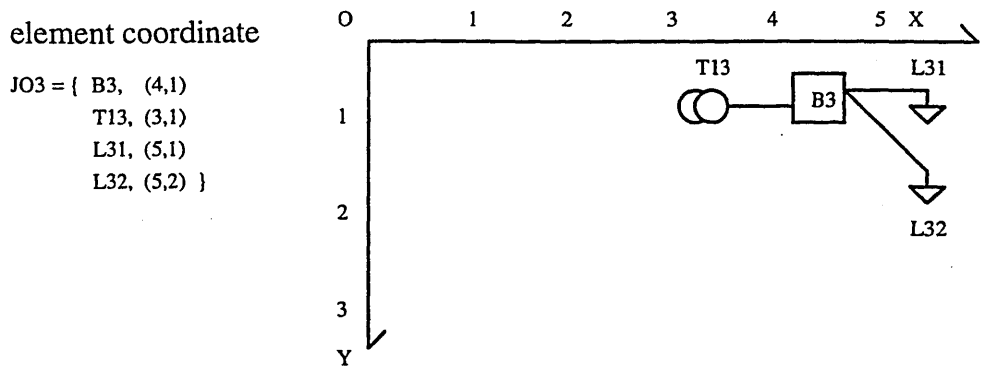


Fig 5.5 Third recursion in diagram generation

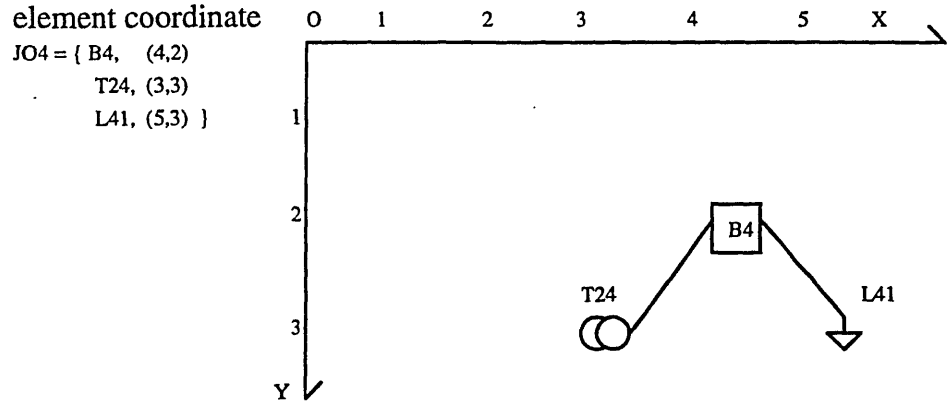


Fig 5.6 Fourth recursion in diagram generation

After the operation for power system network element placement, the data for the connection lines are generated by the ANDG program. The rule is to use the B element in JO as the start point and the G, L, T, C elements as the ends to draw the connection lines. Since each T or C element is a connection element with a unique co-ordinate and is recorded in both JO_i and JO_j ($i \neq j$), it is the connection node for JO_i and JO_j in the network one line diagram.

If the above JO_i (i=1,2,3,4) is drawn on the drawing board, a complete power system network diagram can be obtained as shown in Fig 5.7

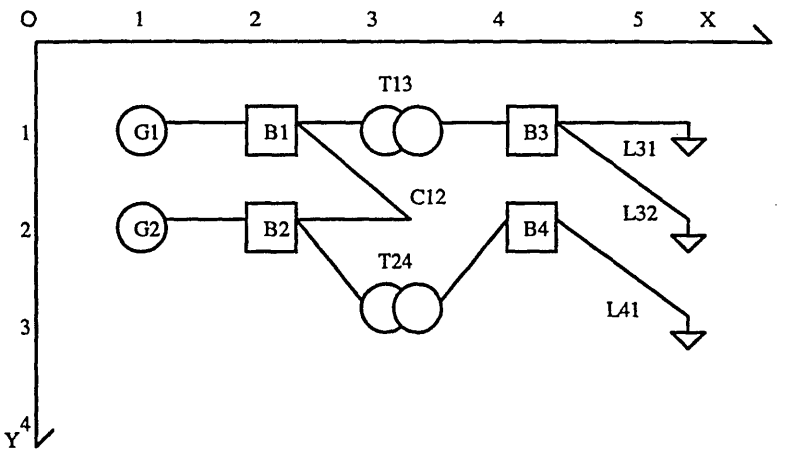


Fig.5.7 Auto generated one line diagram

5.2 CAD styled one line network diagram drawing

As mentioned in Chapter 2, CAD styled power system one line network diagram drawing involves approaches from two different angles:

- drawing electrical schematics, and
- interactive graphics interface for drawing one line diagrams.

The latter can provide users with ASCII files containing the power system network topology description. It has been adapted and improved by this research for the LIME in order to support the drawing of very large scale power system network diagrams.

5.2.1 The concept of three drawing boards

The basic concepts of Drawing Board and Viewport which have been discussed in Chapter 4 (section 4.5), are further developed in this chapter to meet the requirements of diagram information management and data scheduling for the LIME. The development involves the concept of three drawing boards with two sets of co-ordinate systems.

The three drawing boards are:

- Background Drawing Board (BDB)

  BDB is a virtual and size-unlimited static diagram projection plane for the LIME, and consists of a set of sub drawing boards. Physically BDB is a data base file recorded on hard disk.

- View port Drawing Board (VDB)

  VDB is the drawing plane corresponding to the display window for the LIME. In LIME operation, VDB is part of the projection of BDB, i.e. BDB can be projected on VDB in part dynamically. Similarly to BDB, VDB is composed of a set of sub drawing boards. VDB physically is a set of buffers in the internal memory.

- Sub Drawing Board (SDB)

  SDB is a small rectangle for drawing element E_i (i=1,2,...) of the power system network and is a basic unit of BDB and VDB. SDB physically is a set of records in the data base file of BDB_i and will be copied to a corresponding buffer when it is called in to system operation. Thus VDB is composed of the buffers of these activated SDBs.

There are two sets of co-ordinate systems employed to describe the position of element E_i (ex_i, ey_i) (i=1,2,...) in the network diagram. These are offset co-ordinates and segment co-ordinates. The offset co-ordinates are used to describe the position of E_i in the SDB, and the segment co-ordinates are used to describe the position of the SDB in the BDB and the VDB. Suppose that E_i's offset co-ordinate is (sx, sy), and SDB's segment co-ordinate origins in BDB and VDB are (bx, by), (vx, vy) respectively, then, as shown in Fig.5.8, the position of E_i in BDB is (sx+bx, sy+by), and in VDB is (sx+vx, sy+vy) as illustrated in Fig.5.9.

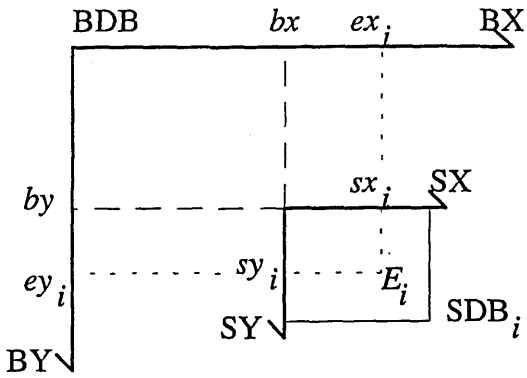


Fig. 5.8 Offset & segment co-ordinates on BDB

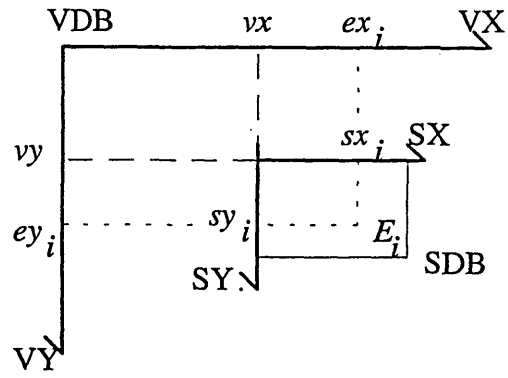


Fig. 5.9 Offset & segment co-ordinates on VDB

If (Δx , Δy) is used to present the differences between these two sets of co-ordinates, then

$$\Delta x = (sx+bx) - (sx+vx) = bx-vx$$

$$\Delta y = (sy+by) - (sy+vy) = by-vy$$

It is easy to see that the change only depends on the segment co-ordinates (bx, by) and (vx , vy). This means that in the process of projecting BDB on VDB every element E_i (i=1,2,...) will keep its unique offset co-ordinates in SDB unchanged. Therefore each SDB can be used as a diagram data carrier in the exchange of data between BDB and VDB, as shown in Fig. 5.10.

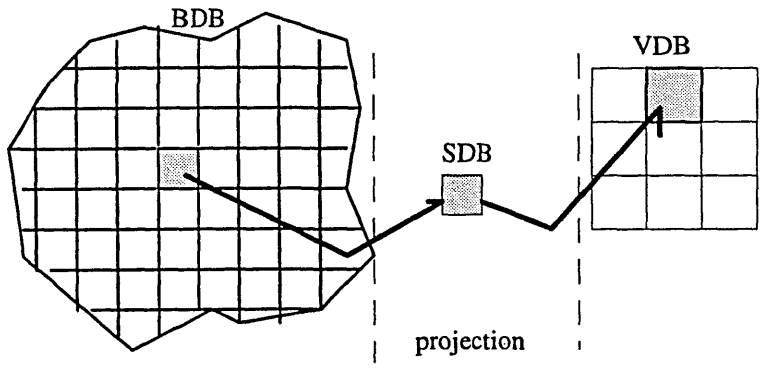


Fig. 5.10 The data carrier in data exchanges

5.2.2 The concept of mapping and projection matrices

Considering that the changing position of E_i (i =1,2,...) is determined by the segment co-ordinates in the process of the diagram projection, the concept of two mapping matrices is used to index SDB data and simplify the segment co-ordinate calculation required for the projection. They are :

- BDB mapping matrix (BM)

As mentioned previously BDB is composed of a set of SDBs. Considering each SDB as an element, the placement of SDB on BDB can be described by its row position i(i=1,2,..., n) and its column position j (j =1,2,..., m) when SDB is activated. Fig 5.11 gives an illustration in the case if all the SDBs are activated.

Column 1	2	... ...	m	Row
SDB 11	SDB 12	... ...	SDB 1 m	1
SDB 21	SDB 22	... ...	SDB 2 m	2
⋮	⋮		⋮	⋮
SDB n 1	SDB n 2	... ...	SDB nm	n

Fig. 5.11 The order of the SDBs in the BDB

So a sparse matrix BM can be obtained, called BDB mapping matrix, and BM=$\{b_{ij}\}$ (i=1,2,...,n; j=1,2,...,m), where b_{ij} stands for SDB_{ij} in BDB. Thus when SDB_{ij} ($i \in \{1,2,...,n\}$; $j \in \{1,2,..., m\}$) is placed in the BDB, its position description (i, j) will be recorded in the database. Also (i, j) is the keyword stating that SDB_{ij} is the unique owner of this position.

- Projection matrix PM

In LIME operation, PM will be automatically created in the internal memory by the system program and then it will keep an active status during the system working process. PM includes two different kind of components. One is, similar to BM , a VDB mapping matrix VM=$\{v_{ij}\}$, and the other is a set of duplex pointers p_{ij}. The following is an example:

$$PM = \left(\begin{array}{ccc:c} v_{00} & v_{01} & v_{02} & p_{03} \\ v_{10} & v_{11} & v_{12} & p_{13} \\ v_{20} & v_{21} & v_{22} & p_{23} \\ \hdashline p_{30} & p_{31} & p_{32} & p_{33} \end{array}\right)$$

where $\{v_{ij}\}$ (i=0,1,2, j=0,1,2) is initialised as an unordered pointer array. This array matches the placement of SDBs in VDB, as shown in Fig.5.12

VM

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

VDB

SDB1	SDB2	SDB3
SDB4	SDB5	SDB6
SDB7	SDB8	SDB9

Fig.5.12 VM and VDB

Here SDB_q (q=1,2,...9) in the VDB is a projection of SDB_{ij} ($i \in \{1,2,...\}, j \in \{1,2,...\}$) in the BDB.

For the data management of the VDB, the corresponding buffer and segment co-ordinate SC (x, y) are created by the system program for each SDB_q (q= 1,2,...,9) in the VDB. These are:

Buffer $[v_{ij}]$

SC $[v_{ij}].x$

SC $[v_{ij}].y$

where v_{ij} =1,2,3,...,9

In LIME operation when a VDB requires an SDB in the BDB, the data carried by the SDB will be written to the corresponding Buffer $[v_{ij}]$ (Suppose that the SDB's position in VDB is matched by v_{ij} in VM). In the process of drawing a diagram, the element E_p (x_p, y_p) of the diagram is placed via its segment co-ordinate SC(x, y) and the data in the corresponding Buffer $[v_{ij}]$ as follows:

$$x_p = \text{SC } [v_{ij}].x + \text{Buffer } [v_{ij}+ p].x$$

$$y_p = \text{SC } [v_{ij}].y + \text{Buffer } [v_{ij} + p].y$$

where p=1,2,3,... .

In the PM, $\{p_{i3}\}(i = 0,1,2)$ and $\{p_{3j}\}(j = 0,1,2)$ is a set of duplex pointers. For the VDB, i of p_{i3} designates the row subscript of the element in VM; for the BDB, the value of p_{i3} is the row subscript of the elements in BM . Similarly, j of p_{3j} and the value of p_{3j} are used for the column subscripts of the elements in VM and BM respectively. p_{i3} and p_{3j} $(i = 0,1,2, j = 0,1,2)$ can be updated dynamically to drive the VDB to change its projection on BDB, where p_{i3} and p_{3j} should satisfy the following conditions:

$$p_{03} = I, \quad p_{13} = I+1, \quad p_{23} = I+2, \quad I = 1,2,3,...;$$

$$p_{30} = J, \quad p_{31} = J+1, \quad p_{32} = J+2, \quad J = 1,2,3,... .$$

This is to keep the space indexed by the BM continuously maintained on the BDB.

When a VDB asks the BDB for an SDB , first , p_{i3} and p_{3j} $(i , j = 0,1,2)$ are combined as the index keyword $K(p_{i3}, p_{3j})$, then , as a message, $K(p_{i3}, p_{3j})$ is sent to the BDB. According to K the matched SDB is searched for in the data base of BDB ,

and then the BDB sends back either the data of the SDB when the search has been successful or Null in any other case to the VDB. Next the VDB writes the data or Null to Buffer[v_{ij}] which is designated by (i, j) from p_{i3} and p_{3j}. So that the SDB or Null has obtained its segment co-ordinate $SC[v_{ij}]$.x and $SC[v_{ij}]$.y in the VDB. Based on the information contained in Buffer[v_{ij}] the diagram can be drawn in the VDB's window.

5.2.3 The geometric size of the SDB

From a user friendly point of view , the geometric size of the SDB should be smaller than that of the VDB, where the size of the VDB is equal to the size of the display window. Generally speaking there should be at least four SDBs in the VDB as shown is Fig.5.13. The reason is to make users able to observe connection parts between SDB_i and SDB_j $(i \neq j)$ when the diagram moves. Theoretically more SDBs a VDB contains, more smooth the movement will be (see section 5.2.6.1). On the other hand, the diagram moving operation involving more SDBs requires more complex algorithm. In order to enable a reasonable smooth movement and also comparatively less complex algorithm, it has been designed in this work that a VDB contains nine SDBs.

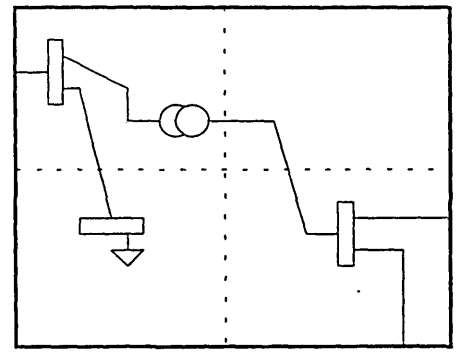


Fig. 5.13 SDBs contained in a VDB

5.2.4 The structure of the Viewport

The basic concept of Projection Matrix(PM) and its role in projecting a diagram from BDB to VDB have been discussed in section 5.2.2. The structure of the Viewport depends directly upon the construction of the PM, and influences the diagram display. To meet the requirement of the three layer drawing board structure and to support a continuously diagram display with screen scrolling, the PM concept has been further extended. The extended PM type matrix PM' has been used in implementing the LIME.

In the electricity supply system, High(H) voltage, Middle(M) voltage and Distribution(D) voltage are three different physical layers. In most existing computer aided systems, there are two methods generally used for drawing power system diagrams. These are

- Single layer drawing board

In a single layer drawing board system, the power one line network diagrams of H, M and D are drawn on a single board, and H, M and D are distinguished by one of the following methods

a) using parameter notes

b) using different colours

c) using different styles of drawing lines

The problem in using a single drawing board is that the number of elements in the different layers are very different. Generally, in any a certain area, the number of elements in the D layer is much larger than that in the M layer, and H layer has the least. In the other words, for a certain number of elements, it covers much a larger

area in the H (or M) layer than in the M (or D) layer. Thus if drawing the network of H, M and D layer in a single board in a scale that the elements of the D layer could be drawn in full detail, then users would not be able to have a global overview of M or H layer through the display window as elements in M and H layer will obviously overflow the screen. On the other hand, drawing the one line diagram in a scale that will allow an overview of the H (or M) layer network, it will not leave enough space to detail the M (or D) layer elements. It is also difficult to analyse and manage the relationship of supply and load among these three layers.

- Using a sub network

The basic concept of this method is to draw the diagram in the proportion allowing an overview of H(or M) layer, and divide M(or D) layer network into a number of sub networks in relation to each element of H(or M) layer. Information about a M(or D) layer sub network can then be displayed by specifying its related element in H(or M) layer. Here each element in H(or M) layer actually is a connection node to its related sub network of the M(or D) layer. This connection is set by registering a pointer of each sub network to the connection node between H and M or between M and D, as illustrated in Fig.5.14.

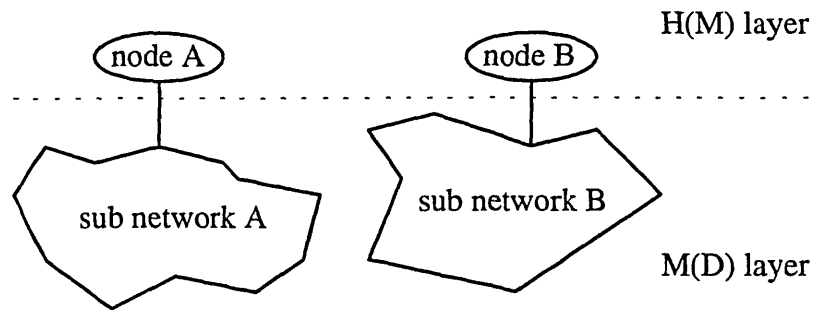


Fig. 5.14 The concept of sub network

Each of these pointers is for indexing the data file of a specific sub network. When the user asks for a connection node, the system will open a window to display the sub network indexed by the pointer registered in that node. This method may overcome the difficulties caused by the mismatching among different layers concerning the areas covered and elements contained. The problem, however, is that this method can support only a network with a tree-style structure (from upper layer to lower layer). It fails to support any other network structures with more complicated connection characteristics such as the one shown in Fig.5.15

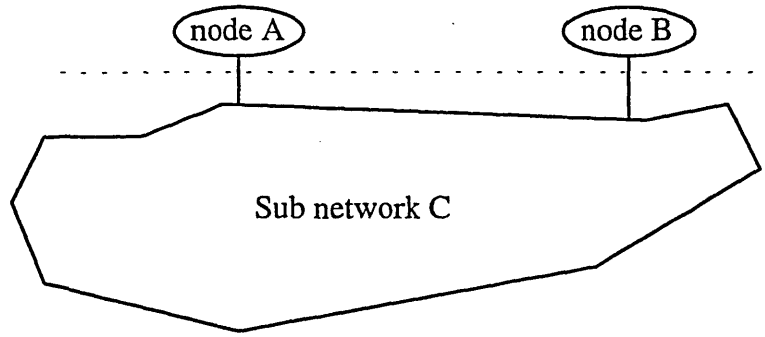


Fig. 5.15 More complicated network structure

To solve the above problems, the new method developed by this research is to provide LIME with a novel drawing environment. In this environment three layer drawing board resources, which are in a proper proportion, are available to the user. The board of each layer is a continuous drawing plane consisting of sub drawing boards. Among these three drawing boards, each SDB in the upper layer is corresponding to a drawing plane, which is composed of a 3×3 SDB set, in the lower adjacent layer. Thus the user can draw the power system network diagram of H , M and D layer using an identical proportion in the different layer. When the power system diagram is being displayed,

Zoom In / Out operation can automatically bring about the changes of the projection proportion between the different drawing boards to keep a proper projection.

In fact the drawing boards are transparent, the three drawing boards can be overlapped in the display for the users to observe the network structure and diagram of M and D layers through the H layer. All these three drawing boards are size-unlimited logical planes, therefore the users are allowed to draw network diagrams without size limitation in any layer.

5.2.5 The data structure of the extended Projection Matrices

The LIME provides three layer drawing board resources, so there are three projection matrices PM_h, PM_m, PM_d needed for managing the corresponding VDB_h, VDB_m, VDB_d respectively, where PM_h stands for PM in H layer, VDB_h stands for VDB in H layer, similarly for the others. The structures of PM_h, PM_m and PM_d are the same. So PM_x ($x = h, m, d$) can be used for the following explanation.

The data structures of PM_x and the corresponding VDB_x (x=h, m, v) are as following

$$PM_x = \begin{pmatrix} x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & x_{06} \\ x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\ x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\ x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\ x_{40} & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\ x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\ x_{60} & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} \end{pmatrix}$$

VDB_x

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

In the PM_x, the VDB_x mapping matrix VM_x (x=h, m, d) is included as a sub matrix of the PM_x. VDB_x is an extended VDB , which contains two zones

a) window edge zone

b) window visible zone

In the VM_x, the distribution of the SDBs within the VDB_x window edge zone is determined by

x_{i1} (i = 1,2,3,4,5), or

x_{1j} (j = 1,2,3,4,5), or

x_{i5} (i = 1,2,3,4,5), or

x_{5j} (j = 1,2,3,4,5).

The distribution of the SDBs within the VDB_x window visible zone is determined by

x_{ij} (i,j = 2,3,4).

The reason for constructing these zones is that the SDBs forms the basis of network diagrams. Without the support of the window edge zone in the display process when the elements of SDB_i have connections with the elements of SDB_j ($j \neq i$) in the diagram (Fig.5.16) , some information may be omitted. This will cause a uncompleted power system diagram, an example of which is given in Fig.5.17. When the window edge zone is employed to support the display, the above problem is solved (see Fig.5.18). This is a particular problem caused by the use of the SDB structure.

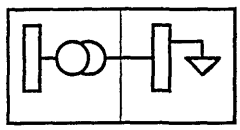


Fig.5.16 Diagram on two adjacent SDBs

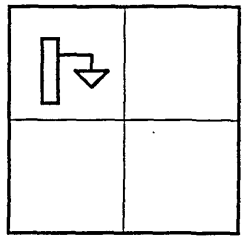


Fig.5.17 Information omitted

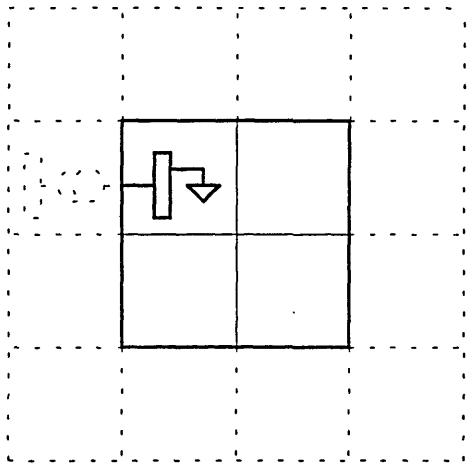


Fig.5.18 Diagram display with window edge zone support

Similar to that mentioned in section 5.2.2, each element of the VM_x corresponds to the distribution of the SDBs in the VDB_x, and is directed at its Buffer [v_{ij}], SC [v_{ij}].x and SC[v_{ij}].y as shown in Fig.5.19

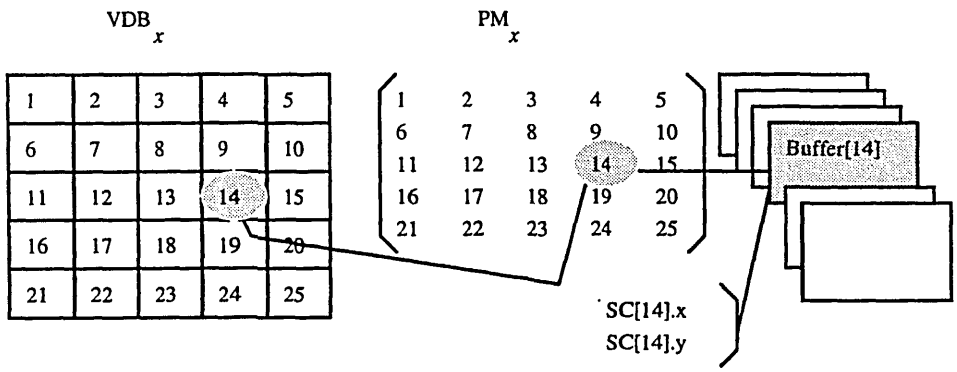


Fig.5.19. The data structure of PM and Buffers

In the environment provided by the LIME, the user is allowed to change the operating layer via the options of Zoom In / Out, i.e. from current focus SDB in any certain layer to its corresponding sub drawing boards in any other layer. Providing this facility requires the system program, firstly to identify the SDB_x ($x \in \{h, m, d\}$) which is the current operating focus, then to calculate the distribution of the corresponding SDB_y ($y \neq x$, $y \in \{h, m, d\}$) in the BDB_x . Thus it is necessary to add x_{0j} (j = 1,2,3,4,5) and x_{i0} (i = 1,2,3,4,5) to the PM_x, where

$$x_{0j} = \begin{cases} 1, & \text{when the corresponding SDB of } v_{ij} \in VM_x \subseteq PM_x, \\ & i \in \{1,2,3,4,5,\} \text{ is activated;} \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{j0} = \begin{cases} 1, & \text{when the corresponding SDB of } v_{ij} \in VM_x \subseteq PM_x, \\ & j \in \{1,2,3,4,5,\} \text{ is activated;} \\ 0, & \text{otherwise.} \end{cases}$$

And also there are x_{i6} (i = 1,2,3,4,5) and x_{6j} (j = 1,2,3,4,5) in the PM_x. These are the duplex pointers between the PM_x and the BM_x, and are used in the same way as those mentioned in the section 5.2.2.

So the extended PM_x includes all the data pointers and the data status information, and it can be used by the system program to schedule and manage data flow and the operation of the projections.

5.2.6 Data Management and Scheduling

In LIME operation, the PM_x ($x = h, m, d$) keeps being active as a data resource which can be shared. The operations that PM_x can support include :

- projection moving

- Zoom In and Zoom Out
- layered drawing boards synchronous moving
- projection transform
- record updating
- parameter data base enquiring

These supports are available for various independent procedures.

5.2.6.1 The projection moving operation

This operation is composed of the following four independent procedures:

- MoveUp,
- MoveDown,
- MoveLeft,
- MoveRight.

Each of these 'Move- ' includes three steps:

(1) move the projection position of the VDB_x ($x = h, m, d$) on the BDB_x

(2) move the SDBs included in the VDB_x

(3) request the BDB_x to obtain the SDBs required to supplement the VDB_x when some SDBs are moved out of the VDB_x

The first step is based on updating of the duplex pointers in the PM_x. This updating changes the position of the PM_x in the BM_x, therefore the project position of the VDB_x on the BDB_x is changed. When the above position is changed, the SDB_i (i = 1,2,...) in the BDB_x also needs a corresponding updating as the second step. This involves serially moving the SDB_i data in the correspond Buffer[p] . Fig 5.20 gives an illustration.

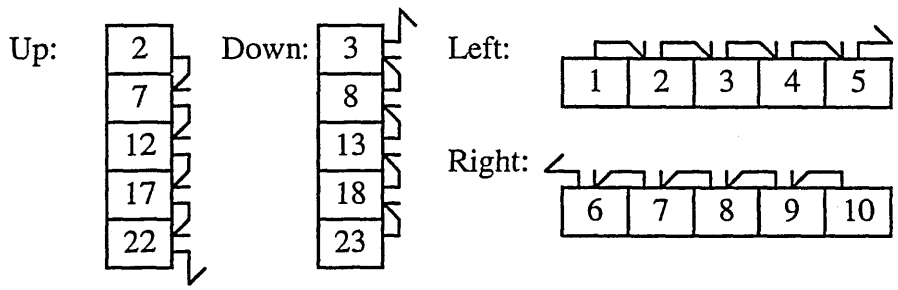


Fig. 5.20 Serially moving the SDB_i data

After the second step , the positions of the SDBs which are still in the VDB_x are changed and assigned with new segment co-ordinates. And the empty buffers caused by moving will be filled by the data of the supplementary SDBs derived from the BDB_x. This is the purpose of the third step.

It could be seen that in each projection moving process, only 1/5 of the data is renewed, and the other 4/5 data is reused in the internal memory. So this method can effectively reduce the costs of the system communication and the time of visiting the hard disk, therefore increasing the efficiency of the system.

5.2.6.2 Diagram Zoom In / Out

This operation includes the following four independent options:

- Zoom In from H to M layer,
- Zoom In from M to D layer,
- Zoom Out from M to H layer,
- Zoom Out from D to M layer.

In the LIME, the scale proportion between the H layer , M layer and D layer is H:M:D = 1:3:9 as shown in Fig. 5.21

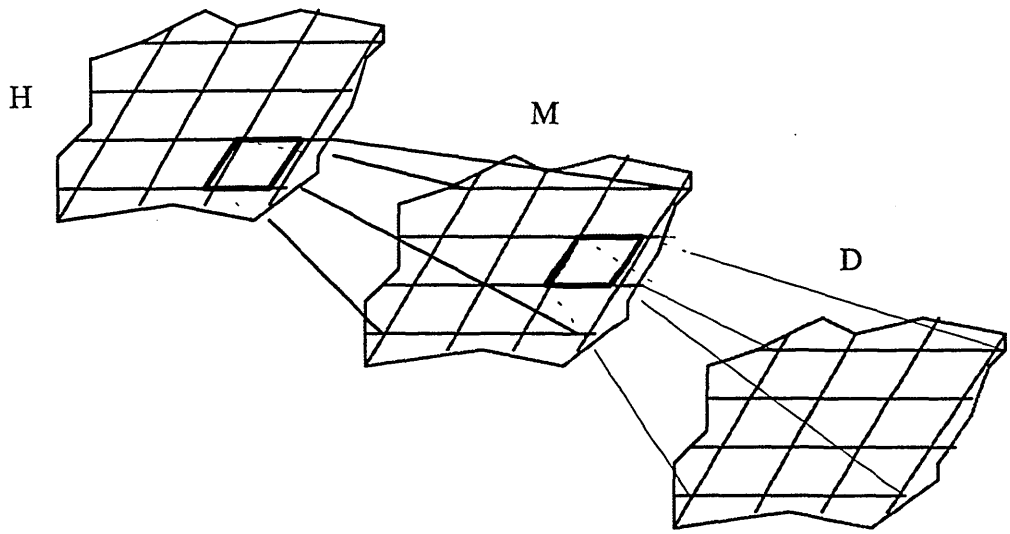


Fig. 5.21 The proportion of H, M and D layer

When the user is in the 'H to M Zoom In' operation the offset co-ordinate of the current focus SDB in H layer will be enlarged three times to match the plane composed by nine SDBs in M layer. This is illustrated in Fig. 5.22

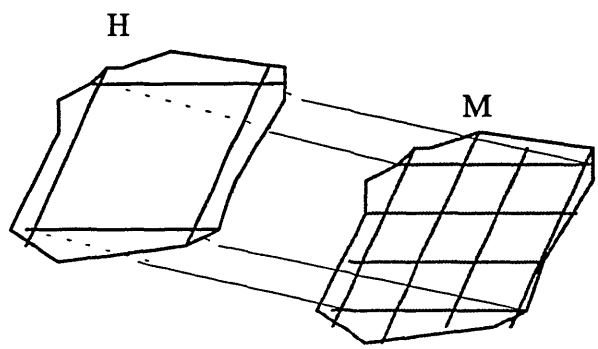


Fig. 5.22 An SDB in the H layer is nine times the size of SDBs in the M layer

When the user chooses the Zoom In option from the M layer to the D layer, the changes of the offset co-ordinate of the SDB in the M layer is similar to the above. Also in this case the offset co-ordinate of the SDB in the H layer needs to be enlarged three times again, i.e. it is nine times the original co-ordinates, to match the plane composed of 81 SDBs in the D layer as shown in Fig. 5.23

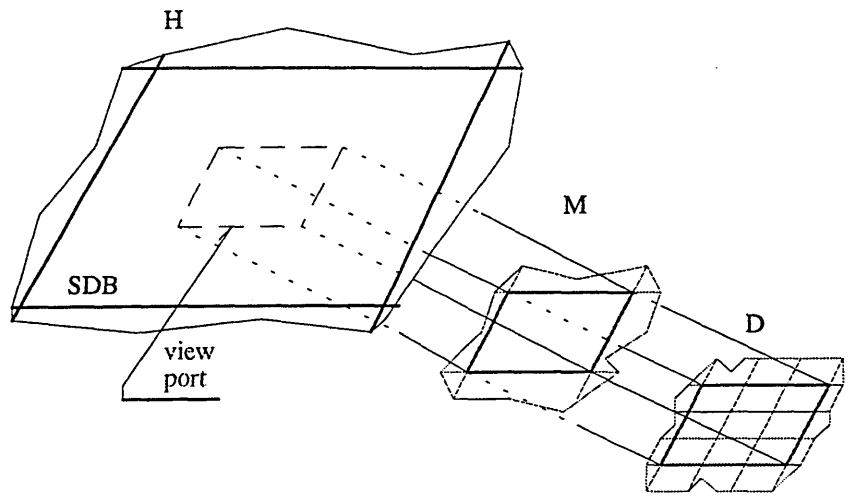


Fig. 5.23 An SDB in the H layer is 81 times the size of the SDBs in the D layer

The Zoom Out operation is a similar concept and is based on calculations in reverse of those in the Zoom In operation.

The operation of Zoom In / Out consists of the following three procedures:

(1) normalise co-ordinates;

(2) calculate the projection subscripts of the SDBs of the goal layer in the BM via the subscript of the focus SDB of the current layer in the PM , and initialise the duplex pointers of the goal layer;

(3) load in the data of the SDBs for the goal layer according to the duplex pointers of the goal layer.

In the system window visible zone, nine of the SDBs in the current layer can be displayed concurrently. The user can choose one of them as the focus for input or observation. Zoom In / Out is performed in accordance with the calculation of the projection subscript of the focus SDB. For normalising the Zoom In / Out operation,

the first step is to remove the focus SDB to the central area of the current layer, i.e. the position corresponding to x_{33} in PM_x . This procedure involves a set of SDBs projection moving operations. Based on this step, the calculation of the projection subscripts of the SDBs in the goal layer can be executed in a normalised state .

5.2.6.3 Moving and display of the layered drawing boards

To concurrently display the layered drawing boards, the system must provide different co-ordinate systems to insure a well proportioned overlap of the projections. These co-ordinate systems are built for the different layers, and the co-ordinates are calculated according to the layer state counter l, where

$$l=\begin{cases}0, & \text{when the drawing board is at the current operation layer;}\\ l+1, & \text{when a Zoom In operation is required;}\\ l-1, & \text{when a Zoom Out operation is required.}\end{cases}$$

In the case of l=0 , the drawing board is the currently active board and can perform the input and update operation required by the user and the co-ordinate system of the currently active drawing board is normalised. When l<0, the drawing board is in a sleeping status, called a sleeping board. The sleeping board is not involved in any system activity and is not shown on the screen. For $l > 0$, the l presents the display layer of the drawing board. Furthermore l is used as the parameter in the updating of the co-ordinates, i.e. the offset co-ordinate of the power system element E_i (x_i, y_i) (i=1,2,...) in VDB_x ($x \in \{h, m, d\}$) is updated according to the formulae below:

$$x'_i = 3^l x_{vq} + 3^l x_i - 3^l x_{v3} = 3^l (x_{vq} + x_i - x_{v3}),$$
$$y'_i = 3^l y_{vq} + 3^l y_i - 3^l y_{v3} = 3^l (y_{vq} + y_i - y_{v3});$$

where

$(3^l x_{vq}, 3^l y_{vq})$ is the segment co-ordinate of the SDB_q (q=1,2,...) in the VDB_x ($x \in$ {h, m, d}),

$(3^l x_{v3}, 3^l y_{v3})$ is the segment co-ordinate of the central SDB of the VDB_x, and is used as the origins updating parameter for the SDBs in the VDB_x during the layer changing process.

In the concurrent proportional display of the layered drawing boards, the projection in moving takes the segments of the current drawing board as the unit moving steps. To keep a correct proportion of the boards in three layers, the unit moving step needs to be projected to, for example , VDB_h and VDB_m . This projection rule is calculated by

$$\frac{3^0(x_{vq}+x_i-x_{v3})}{3^l(x_{vq}+x_i-x_{v3})}=\frac{1}{3^l}$$

$$\frac{3^0(y_{vq}+y_i-y_{v3})}{3^l(y_{vq}+y_i-y_{v3})}=\frac{1}{3^l} \qquad (l=0,1,2)$$

This means when the current drawing board is moved n units, element E_i (x_i, y_i) (i=1,2, ...) in any of the SDBs in the other upper drawing board(s) should have a corresponding move according to:

$$x_i' = x_i(1 \pm \frac{n}{3^l})$$
$$y_i' = y_i(1 \pm \frac{n}{3^l})$$

To manage the moving of the three layer drawing boards, two diagram moving counters D_x and D_y are used in the system for each VDB_x ($x = h, m, d$), where

$$D_x = \begin{cases} D_x + 1, & \text{when diagram is moving left;} \\ D_x - 1, & \text{when diagram is moving right.} \end{cases}$$

$$D_y = \begin{cases} D_y + 1, & \text{when diagram is moving up;} \\ D_y - 1, & \text{when diagram is moving down.} \end{cases}$$

So the actual updating needed for the power system element E_i (x_i, y_i) in VDB_x is calculated by the following formula:

$$x'_i = 3^l(x_{vq} + x_i - x_{v3})(1 + \frac{D_x}{3^i}) = (x_{vq} + x_i - x_{v3})(3^l + D_x)$$

$$y'_i = 3^l(y_{vq} + y_i - y_{v3})(1 + \frac{D_y}{3^l}) = (y_{vq} + y_i - y_{v3})(3^l + D_y)$$

where $D_x/3^l$ and $D_y/3^l$ describe the proportion of co-ordinate transform in the layer designated by l. When $D_x/3^l$ = 1 or $D_y/3^l$ =1, which means the distance of the co-ordinate transform is equal to the segment co-ordinate x_{vq} or y_{vq} , the SDBs in the current drawing board should have a segment co-ordinate transform. So D_x and D_y is a one-digit number counter, where the carry is 3^l. It means when $D_x = 3^l$ or $D_y = 3^l$ the carry is functioned and then $D_x = 0$ or $D_y = 0$. This carry is a symbol, which directs the system program to operate the normalised projection moving to the SDBs in the layer l.

5.2.6.4 Add new records to BDB_x

The blank table for storing the data of the BDB_x ($x = h, m, d$) is established when the user creates a set of new data base files using the facilities provided by the LIME. The record of a power system element in the BDB_x table includes information about

- which layer the element belongs to, i.e. the element is in BDB_h or BDB_m or BDB_d;
- which SDB the element belongs to .

The information is written in the first field of the record. This field as the index keyword consists of the following three components

layer	BM_c	BM_r

where

$$\text{layer} = \begin{cases} \text{'H', one digit} \\ \text{'M', one digit} \\ \text{'D', one digit} \end{cases} ;$$

BM_C presents the element column subscript in BM, four digit;

BM_r presents the element row subscript in BM, four digit.

So a BDB_x table for storing three layer drawing boards is designed to be able to manage $3 \times 9999 \times 9999$ SDBs. And each SDB buffer allocated by the system has a capacity of nine records. Thus the maximum number of power system elements which can be managed by one BDB_x table is

$$9 \times (3 \times 9999 \times 9999) = 2699430327 \approx 2.7 \times 10^9 .$$

In the procedure of adding new records to BDB_x the index keyword is automatically written by the system. To perform this task, the variable 'activelayer' for current layer management is set in the program, where

$$\text{activelayer} = \begin{cases} \text{'H', when H is the current layer;} \\ \text{'M', when M is the current layer;} \\ \text{'D', when D is the current layer.} \end{cases}$$

It is used to record the current active layer chosen by the user.

The process of adding a new record to the BDB_x includes the following six steps:

(1) the user chooses the element of the power system , and the chosen object is the current active object ;

(2) the user designates the element position in the VDB_x via the use of the mouse;

(3) according to the position designated by the user and the segment co-ordinate on the VDB_x, the system program identifies the SDB which the designated position belongs to and calls the input procedure of the SDB.

(4) In SDB_q (q= 1,2,...) input procedure , the new element E_i (i=1,2,...) obtains its offset co-ordinate (x_i , y_i) in SDB_q from :

$$x_i = x_{Ei} - x_{vq}$$

$$y_i = y_{Ei} - y_{vq}$$

where (x_{Ei} , y_{Ei}) is the position of E_i in the VDB_x , (x_{vq} ,y_{vq}) is the segment co-ordinate of SDB_q .

(5) combining the row & column subscripts of the SDB in the BM_x with variable 'activelayer' as the keyword , and writing it to the buffer. The above row and column subscripts of the SDB is determined by the corresponding duplex pointers of the SDB in the PM_x .

(6) writing the complete record stored in the buffer to the BDB_x table on the disk .

5.3 Summary

Based on the database data table structures designed in Chapter 4, the work presented in this chapter has concentrated on the methods of drawing one-line network diagram for very large scale power systems. Two different techniques have been discussed, one concerns automated generation and one is CAD styled drawing. The automated generation method has been implemented in a prototype information management system with success in the support of diagram and parameter management for very

large scale power system networks. However, the benefit provided by this method has been restricted to a fast generating process, and some important requirements regarding *User Conceptual Model* for such information management systems have been unsatisfactory. Therefore the CAD styled drawing method has been used in the implementation of the LIME-Power Shell. To allow this method to be used for very large scale power systems, a number of concepts and techniques have been proposed and developed in this research work. These include the concepts of Background Drawing Board, Viewport Drawing Board, Sub Drawing Board, three-layer Drawing Board, segment co-ordinate, off-set co-ordinate system, mapping matrices, and the techniques of data flow scheduling and programmed drawing. The use of these techniques has enabled the LIME to manage one-line network diagrams in a different way from what has been applied traditionally. It has allowed a size-free diagram drawing with parameter management support, which has in the past been restricted by PCs internal memory capacity and was not possible with existing CAD systems. This has, therefore, provided a solution to the key issue in the LIME development regarding its capability of managing diagrams and device parameters for very large scale power systems.

CHAPTER 6

IMPLEMENTATION OF THE LIME-POWER SHELL

In this chapter the implementation work of the LIME, called Power Shell, is discussed. The implementation of the Power Shell program has been based on the data structures designed in Chapters 3 and 4, and follow standard software development steps and methods such as the system analysis method, the top-down design method, and the structured design method. It has also involved a number of programming techniques. These include

- the modular programming method;
- concurrency techniques for the DIMS real time communication requirement;
- Windows techniques for providing standard user interface;
- database techniques and data flow scheduling methods for managing diagrams and parameters for very large scale power system networks;
- data dictionary techniques for allowing flexible data structure definitions;
- and object oriented operation method for supporting a convenient user operation environment.

The Power Shell system has been implemented in MS-Windows, C and Paradox Engine, and consists of about 30,000 lines of code (LOC). It is a running system and has demonstrated the achievement of the original objective.

The design and implementation of the Power Shell program consists mainly of the following steps:

(1) System input analysis;

(2) System output analysis;

(3) Data structure design for data processing between input and output;

(4) User operation procedure analysis;

(5) System characteristics analysis and development tool selection;

(6) Program structure design;

(7) Sub job module design and implementation;

(8) System debugging and testing;

(9) User Guide.

These are detailed sequentially in the following sections except for the debugging and testing. The testing work will be introduced in the next chapter for system evaluation.

6.1 System input analysis

The basic concept for system input analysis is to consider the LIME to be implemented, i.e. the Power Shell system, as a 'black box', and then to list and categorise all the information required for input. The Power Shell program should be designed to manage all these inputs. As the LIME of the DIMS, the input of the Power Shell system involves both data input from the DIMS computer network and from the LIME local users.

Data input from the computer network is mainly the data relating to power system running status. This includes:

- The power system current running status data, which is from the on-line data acquisition computer (SCADA system).
- The power system historical running status data, which is from the database of the industrial host computer.

Data input from the local user includes the following categories:

- Power System Project data

  Purpose: specify Power System Project name

  Input type: character strings

  Input device: keyboard

- Database data structures

  Purpose: redefine data structures of the database tables for power system elements

  Input type: character strings

  Input device: keyboard

- Power system network diagram

  Purpose: specify the co-ordinates of the diagram element symbols on the drawing board;

  specify connection relationships between power system elements.

  Input type: co-ordinate

  Input device: mouse

- Power system element parameters

  Purpose: input parameters of power system elements to database tables

  Input type: character strings

  Input device: keyboard

- Power system network diagram update

  Purpose: specify the co-ordinates of the diagram element symbols on the drawing board;

  specify connection relationships between power system elements.

  Input type: co-ordinate

  Input device: mouse

- Power system element parameter update

  Purpose: input parameters of power system elements to database tables

  Input type: character strings

  Input device: keyboard

- Drawing board

  Purpose: for diagram Zoom In/Out and Move

  Input type: instruction

  Input device: mouse

- ASCII file name

  Purpose: specify data exchange file name

  Input type: character strings

  Input device: keyboard

- Power system application program result

Purpose: read back the result of the power system application program to the database table for displaying the results on the network diagram

Input type: ASCII file

Input device: programmed read-in

- Task scheduling

Purpose: call sub-routine

Input type: instruction

Input device: mouse

- Input error

Purpose: reject error keywords for the Power Shell databases

Input type: N/A

Input device: programmed check

6.2 System output analysis

Similar to the input analysis, considering the Power Shell as a 'black box', the processed results required as outputs have been listed and categorised. The Power Shell program should be implemented to be capable of supplying all these outputs. These contain the outputs required for other computers in the DIMS computer network and for the LIME local users.

Output to other computers in the computer network is mainly the data of the power system network, which can be further detailed by:

Purpose: provide source data for power system application program process on other computers

Output type: ASCII files

Output device: common communication software and the computer network

Local outputs include the following data categories:

- Database table data structure

  Purpose: display database table data structure for user re-definition

  Output type: table

  Output device: Windows

- Power System Project

  Purpose: provide Power System Project information for power system application program process

  Output type: ASCII files

  Output device: programmed generation

- Power system network diagram

  Purpose: display network diagram of Power System Project for user to view and update

  Output type: diagram that user can move and Zoom In/Out

  Output device: Windows

- Parameters of power system elements

  Purpose: display parameter tables of the power system element for user to view and update

  Output type: table

  Output device: Windows

6.3 Data structure for data processing between input and output

In addition to the system input and output analysis, in which the Power Shell system was considered as a 'black box', the analysis here deals with the 'black box' internal data structure for processing input data to produce the required output. The analysis and design of the internal data structure is one of the key steps in the implementation of the Power Shell system. Basically the data structure needs to be well defined to

- record and manage all the input data of Power Shell, and describe the data relationships;

- support data accessing and scheduling required for the Power Shell program;
- give the program a comparatively high running efficiency; and
- support the production of all the required outputs.

The details for the data structure analysis and design are given in Chapter 4.

6.4 User operation procedure analysis

The design and implementation of a large integrated software environment program consisting of several tens of thousands LOC is much more complex and difficult than that of a simple task program containing only a few thousands LOC such as a specific calculation program. This is because of a number of reasons. The first problem is due to the limitation of the computer hardware capacity. This makes it impossible for such a large program to be loaded into the computer at one time. Then a difficulty has also arisen from the software program itself. Generally when a software program is larger than two or three thousands LOC, the debugging will be very difficult. Meanwhile a large job module may require hard work to ensure its stability and reliability.

One common and effective way of solving these problems is to divide such a large software system into a number of independent sub job modules, and then through a job scheduling module to manage all these sub job modules. There are some key points in the design of such sub job modules. The size of each sub job module should not be larger than ten thousand LOC and each sub job module should achieve an entire task. These sub job modules should also be able to co-operate with each other in the integrated task involving several sub job modules. In the design work, these require the detailed analysis and complete understanding of the job service that the Power Shell system should provide for users; then the determination of the necessary sub job modules and the identification of the sub modules that may be involved in the co-operation for some integrated tasks; also the incorporation and scheduling of these sub modules in the Power Shell system.

The following job services are required for the Power Shell program:

- Create a new Power System Project

The Power Shell system is required to manage information for different power system networks and different power system application programs. Each of these tasks will be started with the creation of a new Power System Project. The procedure of creating a new Power System Project involves three user operations:

(1) Create and register a set of new database tables for recording the data input by users. The default structures of these tables are given by Power Shell.

(2) Redefine data structures. Users can redefine these database table structures for their specific requirements.

(3) Input the network diagram and related parameters for the Power System Projects created.

The job flowchart for creating new Power System projects is show in Fig. 6.1

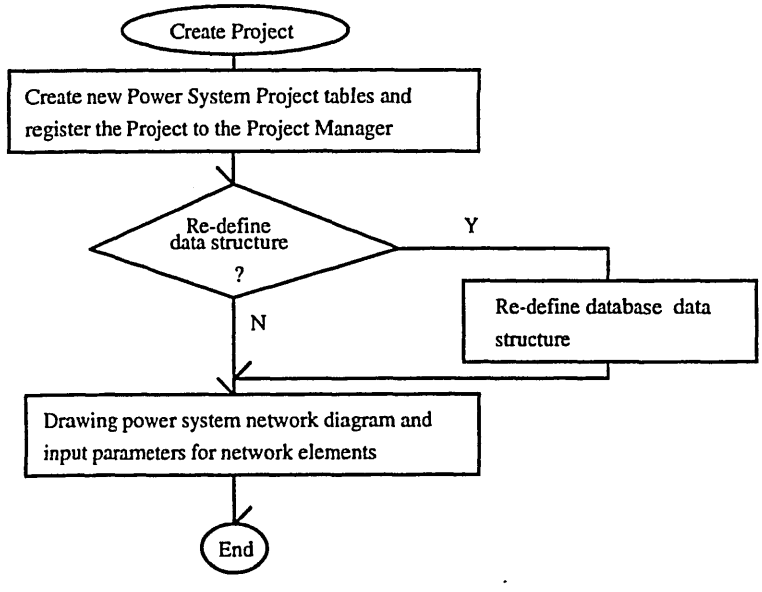


Fig.6.1 Creating a Power System Project job flowchart

• Data structure maintenance

Data structure maintenance is for users to update the structures of the database tables that have been registered in Power Shell. This procedure involves four user operations:

(1) Create and register a set of new database tables.

(2) Redefine database table structures.

(3) Copy the unchanged data from the original tables to the new data tables.

(4) Input additional data to the new data tables.

Fig.6.2 shows the flowchart of the data structure maintenance job.

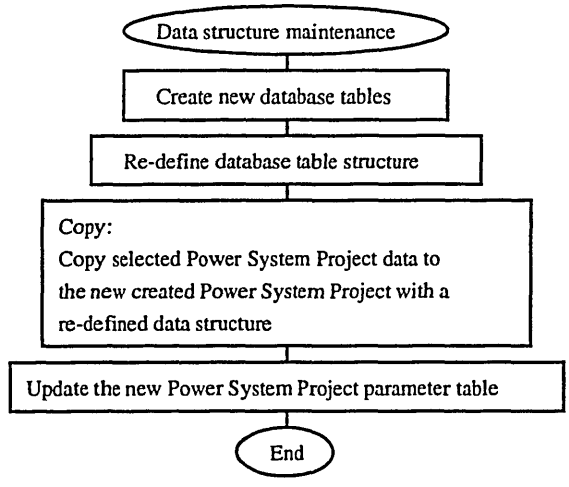


Fig.6.2 Data structure maintenance job flowchart

• Updating diagrams and parameters

This is for users to update a Power System Project diagram and parameters. The user operation is achieved through the Power Shell diagram interface. The process involved is illustrated in Fig.6.3.

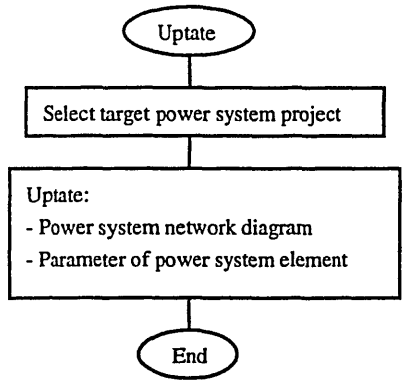


Fig.6.3 Diagram and parameter updating job flowchart

- Operation for power system applications

This operation requires the co-operation of the Power Shell diagram interface, data input and output sub job modules, and the users application program. The job flowchart is given in Fig.6.4.

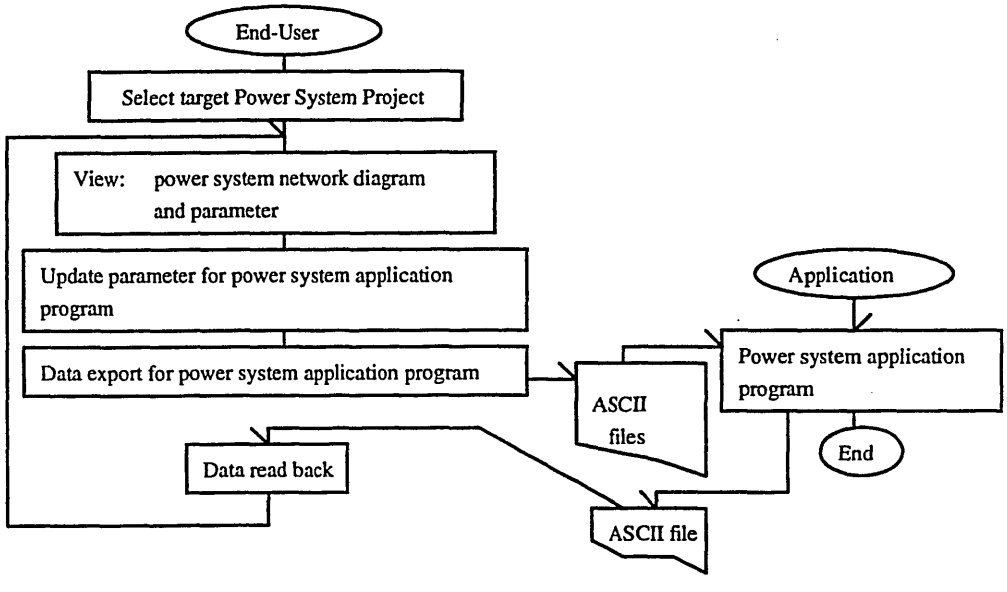


Fig.6.4 Power system application job flowchart

From the above discussion of the operation job service required for Power Shell, it has been found that these services have some sub operations in common. For the optimum Power Shell system structure, these common sub operations can be developed to independent sub job modules, and then complex tasks can be achieved by the

combinative use of these sub job modules. Therefore the Power Shell is composed of the following 9 sub job modules for:

- job scheduling,
- Power System Project management,
- diagram operating,
- parameter table operating,
- creating Power System Project table,
- data table structure redefinition,
- Power System Project copy,
- data input, and
- data output.

6.5 System characteristics analysis and development tool selection

Power Shell is designed as the LIME for the local computer platform in the DIMS. Suppose in the DIMS operation the host computer broadcasts the power system running status data periodically, then the Power Shell LIME will be cyclically interrupted by the communication program for releasing the CPU to receive data and to make a response. In the current computer hardware and software techniques, the communication is actually supported by the hardware interface, e.g. RS232. The major task of the RS232 is to receive and send data sets. In the process of receiving and sending one set of the data, RS232 works independently and does not occupy a time slice of the CPU. The CPU will be interrupted and requested for communication data processing only if the RS232 asks to exchange data with the CPU. Therefore in the PC computer's communication operation, its CPU need not stop other jobs completely. There is the possibility for the CPU to serve both the communication job and other local jobs in parallel through a time sharing method. This means that the communication program could be designed as a background program running in parallel with other programs. In current PC operating systems, DOS allows only single

job operating and normally cannot support multi-job concurrency. The latest operating system MS-Windows is a time sharing operating system and supports multi-job running in parallel on the PC platforms. So if Power Shell is designed as a MS-Windows application program, it is possible for the Power Shell system to run in parallel with the communication program or other power system application programs. Users can benefit from this multi-job concurrency since the communication program can receive up-to-date data of power system running status from the host computer as the background job, and then the foreground Power Shell program need not be interrupted by the communication requirement.

The selection of the operating system is the basis of developing Power Shell as a multi-job concurrency program. On the current software market, MS-Windows is the most popular operating system supporting multi-job concurrency for PC platforms. MS-Windows has also provided a well constructed interface for standard C. This allows the use of C to call the sub functions of MS-Windows to implement application programs in the standard MS-Window style. Such a standard Windows program will automatically inherit benefits from MS-Windows for its supports on multi-job concurrency and LAN communication services. Furthermore, as mentioned in Chapter 4, the relational database techniques have been used for Power Shell to manage a large amount of data. This requires employing a database system that can connect both C and MS-Windows in the implementation of the Power Shell system. Therefore Borland Paradox Database Engine has been considered. The database tables built in Paradox Engine are also compatible with other relational database systems such as Paradox database, dBaseIV and Foxbase. This will allow data access to the Power Shell database tables built in Paradox Engine from those relational database systems. Thus eventually the following software development tools for the implementation of the Power Shell system have been selected:

- MS-Windows,
- C (Borland), and

- Paradox Engine (Borland).

6.6 Program structure design

Using a modular programming method the Power Shell program is composed of a number of sub job modules. In order to avoid memory overflow in the execution of these sub modules, each sub module of the Power Shell program has been designed to have less than ten thousand LOC. These sub modules are independent of each other, and can be developed and debugged individually. At the top level of the Power Shell program there is a job scheduling module that drives all other sub job modules. The framework of the Power Shell program is illustrated in Fig.6.5.

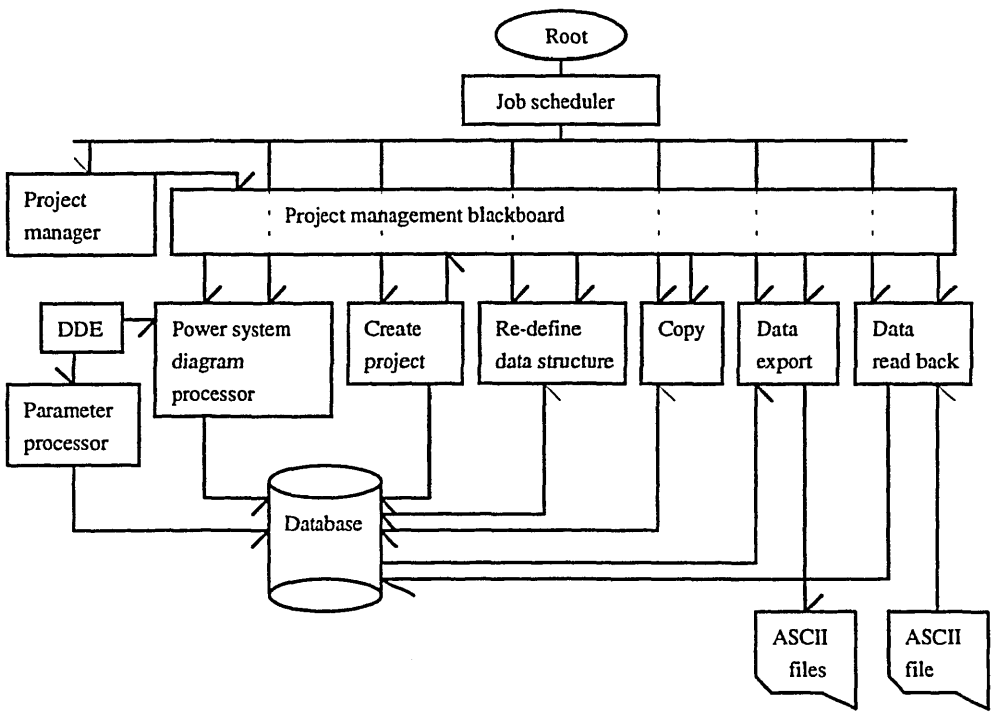


Fig.6.5 Power Shell program framework

Using this modular method, i.e. constructing the system with independent sub modules, will benefit the Power Shell program regarding its maintainability. If, later, Power Shell is required for any new tasks, all that is needed to be done is to implement the new sub module for the required task and then register the new module

to the job scheduling module. Other sub job modules will not be involved in this updating. Similarly, for any improvement required later for the existing functions, only the corresponding module needs to be considered and revised.

6.7 Sub job modules

The Power Shell system is composed of 9 sub job modules that are independent of each other. These are modules for job scheduling (i.e. Job Scheduler), project management(i.e. Project Manager), project creating, diagram drawing, parameter redefinition, project copy, data export, data read back, and parameter table service. The design and implementation of each of these modules is detailed in the following sub sections.

6.7.1 Job Scheduler

The Job Scheduler is the module for managing 7 of the other 8 sub job modules, the parameter table service module being managed by the diagram drawing module. It provides users with the Power Shell root window, shown in Fig.6.6, as the user interface. Users can select job options through the menu provided in the window by using the mouse. There are 7 options available for the user to select, each corresponding to the 7 sub job modules managed by the Job Scheduler.

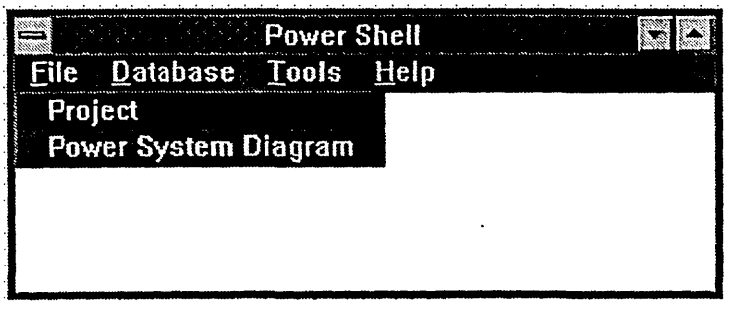


Fig.6.6. Power Shell Job Scheduler

In the process of drawing diagrams, the user can operate the mouse. There are 25 user operations supported by the Drawing Diagram program. The top level program for drawing diagrams is described by the program flowchart shown in Fig.6.14.

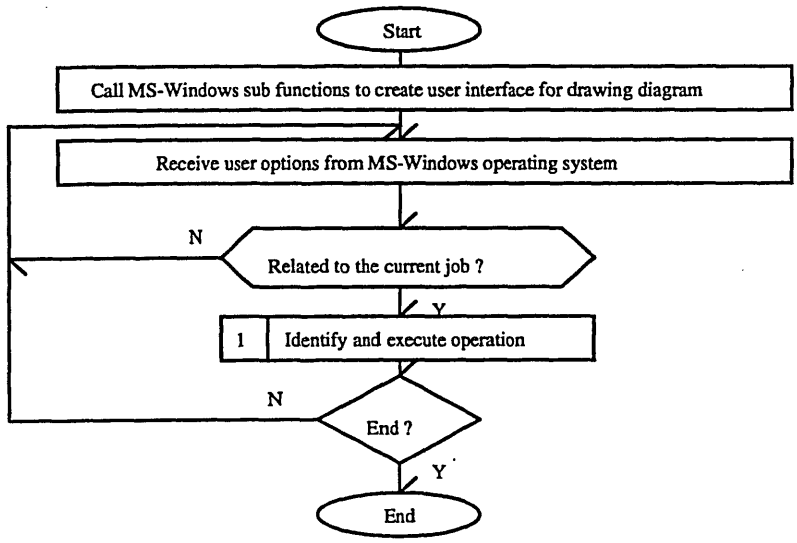


Fig.6.14 The Drawing Diagram program flowchart

The 'identify and execute operation' program can be further described by the flowchart shown in Fig. 6.15.

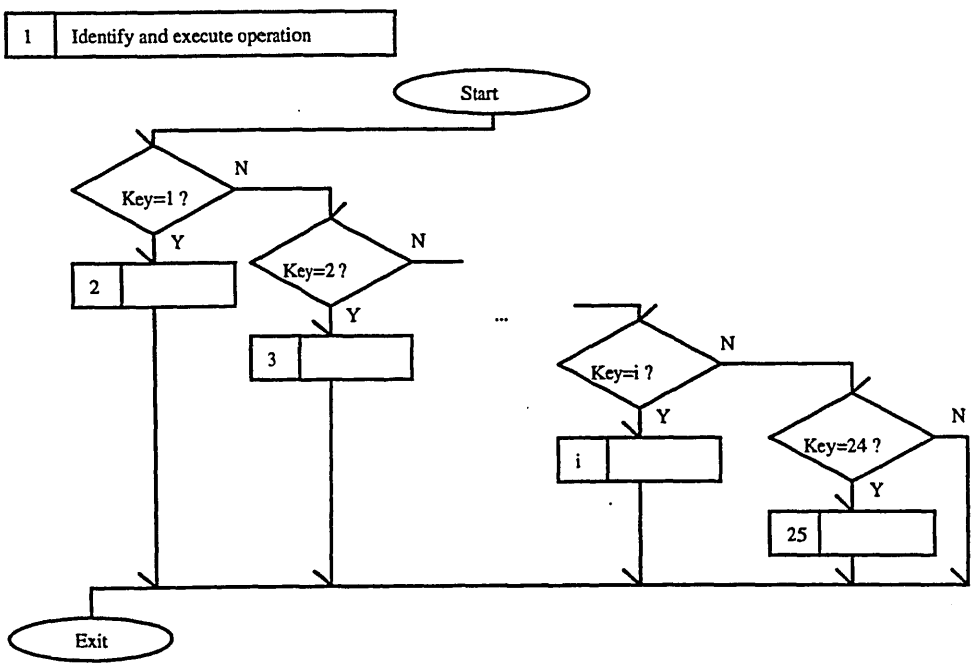


Fig.6.15 Program flowchart for identifying and executing operations

This program must first identify 24 operations and then execute a number of relevant sub functions to achieve the required tasks. These 24 operations are:

2	identify the currently active Power System Project
3	end the drawing program
4	maximise diagram window
5	minimise diagram window
6	move drawing board left
7	move drawing board right
8	move drawing board up
9	move drawing board down
10	designate the current sub drawing board
11	zoom to high layer of drawing board
12	zoom to middle layer of drawing board
13	zoom to low layer of drawing board
14	drawing busbar
15	drawing cable
16	drawing transformer
17	drawing load

18	drawing generator
19	on-line help
20	parameter table operation
21	receive result of parameter table operation
22	cursor location
23	power system network diagram partitioning
24	inquiring parameter
25	data display style operation

These operations are supported by the corresponding sub programs of the Drawing Diagram program. These sub programs may also involve some lower level programs. The following sub sections gives details for the major sub programs.

6.7.4.1 Sub programs

2	identify Current Project

In the Drawing Diagram program operation, the user is required to first open a Current Project, i.e. the currently active Power System Project. Since the Power System is the LIME of the DIMS, the Power Shell database tables may be shared by other application programs or other LIMEs. This means that the Power Shell database table should not be continuously opened for a long time. Each table should be opened only if Power Shell program requires data access, and closed at once when the data accessing process has been finished, to enable other programs to open Power Shell data tables and access data. Therefore the major task of the identifying Current Project program is to find the Current Project name from the Power System Project table and

write it to the buffer for other programs to read the Current Project name and to open corresponding data tables when necessary. Otherwise to read all the tables related to the Current Project may require that these tables stay open for the whole Drawing Diagram process. The process of identifying the Current Project is to open the Power System project table and search the Current Project, and then

if find: register all the data table names of this Project to the target buffer, close the Power System Project table, and return message of success;

otherwise: display information to remind the user to activate a Power System Project, close Power System project table, and return message of failure.

Fig.6.16 shows the identifying Current Project program flowchart.

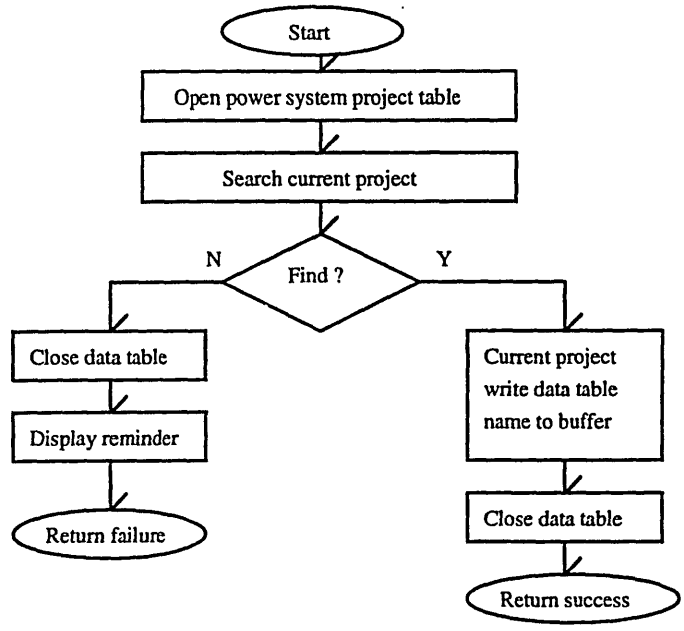


Fig.6.16 Identifying Current Project program flowchart

3	end the drawing program

This program contains the following two tasks:

- to close database operating environment and all the data tables;
- to notify the MS-Windows operating system that the Drawing Diagram operation is ended.

4	maximise diagram window

The 'maximise diagram window' program has been implemented by using MS-Windows functions. There are mainly three tasks:

- to notify MS-Windows to maximise the Drawing Diagram window;
- to call windows refresh function for MS-Windows;
- to call Power Shell 'draw' program, | 26 | draw | (detailed later in section 6.7.4.2)

5	minimise diagram window

Similar to the maximising program, the 'minimise diagram window' program has also been implemented by using MS-Windows functions, which just need to notify MS-Windows to minimise the Drawing Diagram window.

6 (7,8,9)	move drawing board left (right, up, down)

The implementation of moving drawing board left (right, up, down) program has been based on the projection matrices and drawing data buffer, which are defined and described in Chapter 5. Moving the drawing board one step left (right, up, down) is achieved by sequentially updating the data in the drawing data buffers relating to the projection matrices, as shown in Fig.6.17. After the drawing data has moved from the source buffer to the target buffer, the data will automatically inherit the drawing board segment co-ordinates of the target buffer. According to the new co-ordinates the 'draw' program (detailed later in section 6.7.4.2) will then draw the diagram element to the next left sub drawing board. This kind of sub drawing board moving is called *segment co-ordinate moving*.

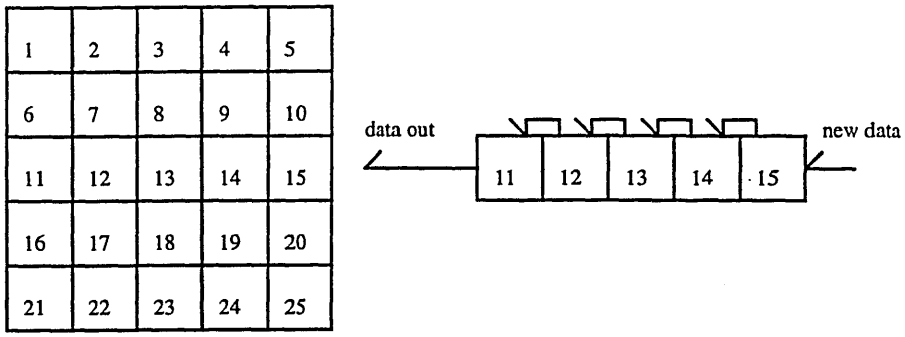


Fig. 6.17 Drawing data sequentially moving

In the circumstances of displaying multi-layers in parallel, the drawing board moving operation becomes very complex. It involves synchronously moving two or three layers in proportion as 1:3:9, calculating co-ordinate transform coefficients, and registering the coefficients to the moving status table. The 'moving drawing board left' program flowchart is given in Fig.6.18.

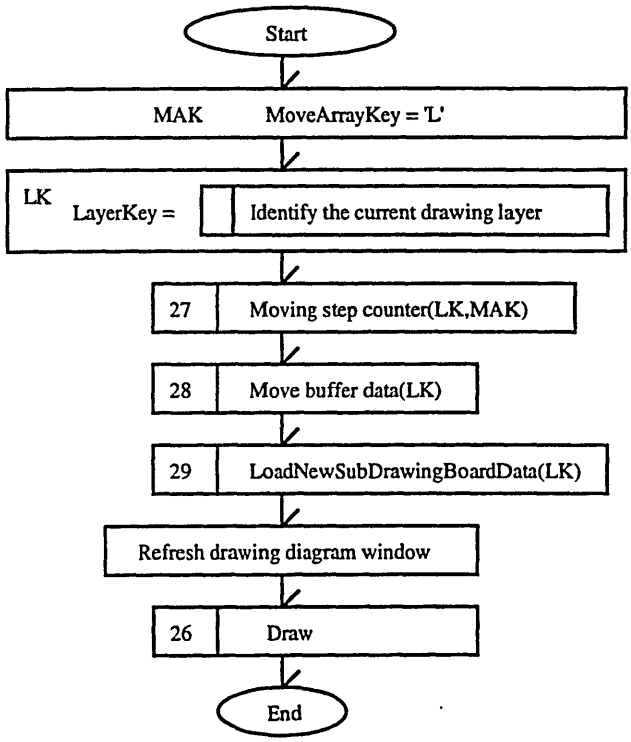


Fig.6.18 Moving Drawing Board Left program flowchart

In the above program, two keywords has been used. They are

- MoveArrayKey(MAK), for specifying the moving direction of the drawing board;
- LayerKey(LK), for specifying the currently active drawing layer.

Both keywords will be transferred to low level sub programs for further drawing work. This Move drawing board left (right, up, down) program also involves some sub programs including Moving step counter, Move buffer data, Load new sub drawing board data and Draw, which are detailed in the next section for Low level sub programs.

10	designate current sub drawing board

The Designate current Sub Drawing Board (SDB) program is a low level program for receiving mouse actions to indicate the SDB where the user's current operation is being carried out. It is required to designate current SDB for 10 sub programs including:

- zoom to H/M/L drawing layer;
- draw a new power system element, i.e. any of B, C, T, L, G or M;
- inquire parameters of the power system elements.

There are quite a few reasons for requiring current SDB identification. For the 'zoom' related programs, the current SDB message is needed for finding out the corresponding SDB on the 'zoom to' layer. The drawing / inquiring power system element programs need to add to /search for data in the corresponding buffer(i) of the SDB which the user is working on, i.e. the current SDB.

The major task of the Designate current SDB program is to mark the SDB the user is currently working on as the current SDB, and the marking is based on the receipt of mouse actions. Fig.6.19 is the flowchart for this program.

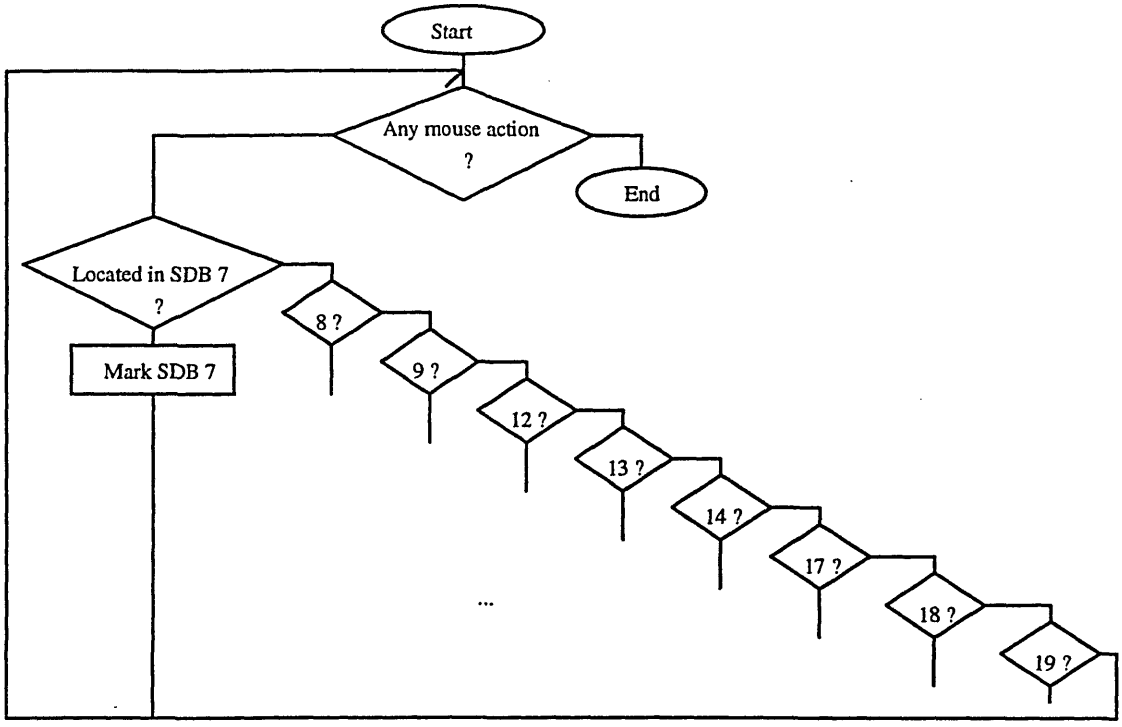


Fig.6.19 The designate current SDB program flowchart

11 (12,13)	zoom to high (middle, low) layer of drawing board

The 'zoom to' operation is a complex program and could have two directions. One is from upper to lower layer, another is the reverse.

- Zoom In to lower layer

As mentioned previously, the View Drawing Board(VDB) of the Power Shell system is composed of several Sub Drawing Boards(SDBs); and the proportion of the H, M and L layer is 1:3:9, as illustrated in Fig.6.20.

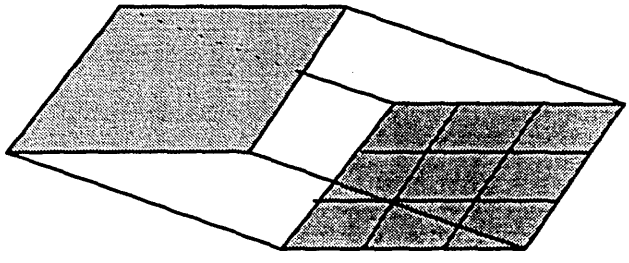


Fig.6.20 The proportion of two adjacent layers

In the process of Zoom In to the lower layer, in order to display all 3×3 SDBs of the lower layer, which are corresponding to the current SDB of the 'zoom from' layer, the first step of Zoom In is to move the current SDB to the centre of the VDB. Fig 6.21 shows an example of moving current SDB to the VDB centre.

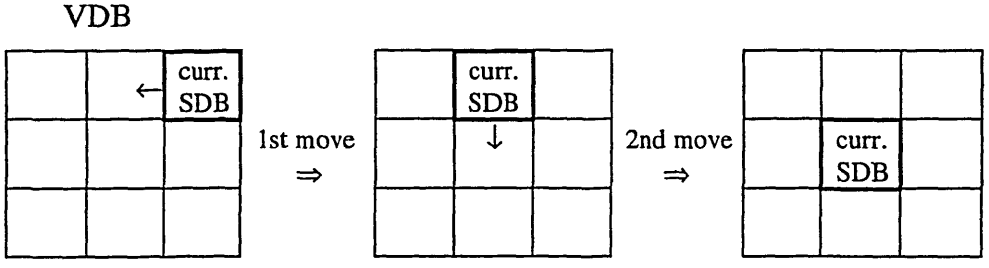


Fig.6.21 The process of moving current SDB to the VDB centre

The second step of Zoom In is to calculate the index keywords of the SDBs of the 'zoom to' layer according to the projection matrix of the 'zoom from' layer, i.e. the upper layer, and write the index keywords as the corresponding duplex pointer to the VDB projection matrix.

Suppose that U_x and U_y stand for the column and row number of the upper layer current SDB in the corresponding VDB projection matrix respectively; the SDBs of the lower layer, i.e. the 'zoom to' layer, are numbered from SDB1 to SDB9, as shown in Fig.6.22; and Si_x and Si_y (i=1,2,...,9) stand for the column and row number of SDBi in the lower layer corresponding VDB projection matrix respectively, then

$$\begin{cases} S1_x = U_x \times 3 - 3 \\ S1_y = U_y \times 3 - 3 \end{cases} \quad \begin{cases} S2_x = U_x \times 3 - 2 \\ S2_y = U_y \times 3 - 3 \end{cases} \quad \begin{cases} S3_x = U_x \times 3 - 1 \\ S3_y = U_y \times 3 - 3 \end{cases}$$

$$\begin{cases} S4_x = U_x \times 3 - 3 \\ S4_y = U_y \times 3 - 2 \end{cases} \quad \begin{cases} S5_x = U_x \times 3 - 2 \\ S5_y = U_y \times 3 - 2 \end{cases} \quad \begin{cases} S6_x = U_x \times 3 - 1 \\ S6_y = U_y \times 3 - 2 \end{cases}$$

$$\begin{cases} S7_x = U_x \times 3 - 3 \\ S7_y = U_y \times 3 - 1 \end{cases} \quad \begin{cases} S8_x = U_x \times 3 - 2 \\ S8_y = U_y \times 3 - 1 \end{cases} \quad \begin{cases} S9_x = U_x \times 3 - 1 \\ S9_y = U_y \times 3 - 1 \end{cases}$$

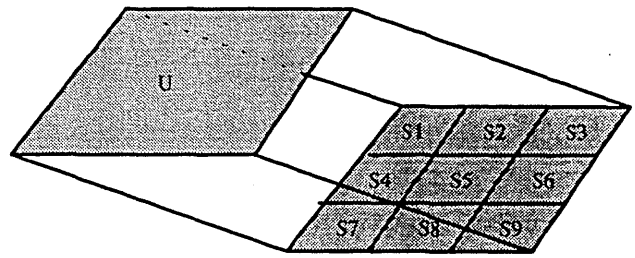


Fig.6.22 Numbered SDBs of the lower layer

The third step is to call sub program 29: Load new SDB data (see section 6.7.4.2 for details) to load the data of the lower layer(i.e. 'zoom to' layer) SDBs to the corresponding buffers. Finally the program calls the Draw program(detailed in section 6.7.4.2) to display the power system network diagram. The program flowchart is given in Fig.6.23.

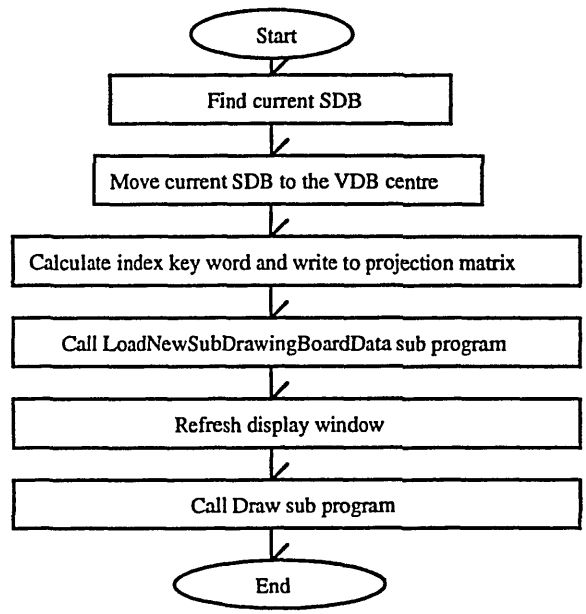


Fig.6.23 Zoom In program flowchart

- Zoom Out to upper layer

This operation is comparatively simple. The program involves clearing the lower layer (i.e. the 'zoom from' layer) SDBs buffers, refreshing the display window and displaying only the upper layer diagram. Fig.6.24 shows Zoom Out program flowchart.

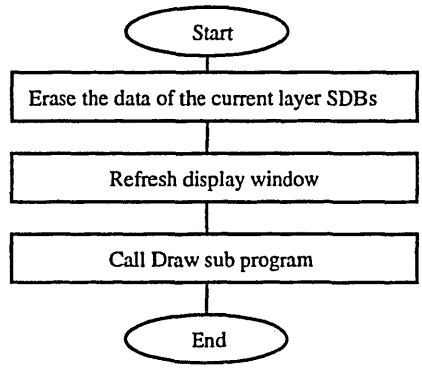


Fig. 6.24 Zoom Out program flowchart

14 /15/16/17/18	drawing busbar/cable/transformer/load/generator

Program 14-18 are similar programs and can generally be called drawing new element programs. Drawing a new power system element requires:

- a NewBuffer for the new element to be drawn;
- the current SDB segment co-ordinate;
- parameter table operating program;
- sub program 26: Draw (detailed in section 6.7.4.2).

To draw a new element involves several sub tasks. The major task is to write the NewBuffer for the new element. The structure of the NewBuffer is the same as the other buffers for recording SDB data. Notified by the mouse action requesting to draw a new element, the program will write the element category(i.e. B, C, T, L, G, or M) to the NewBuffer, then according to the location of each mouse action write the following data to NewBuffer:

- the element starting point co-ordinates,

- the co-ordinates of the first turning point of the connection line,
- the co-ordinates of the first ending point of the connection line,
- the co-ordinates of the second turning point of the connection line, and
- the co-ordinates of the second ending point of the connection line.

After each time of writing co-ordinates to the NewBuffer, the drawing new element program calls the Draw program to draw and display the renewed diagram.

When the process of loading in NewBuffer with the category keyword and corresponding co-ordinates has been completed, the program calls, by way of Dynamic data Exchange, the parameter table operating program to input parameters for the new element. The drawing new element program needs to provide the category of the new element for the parameter operating program to open the corresponding database table for the new element.

Next, after receiving a "success" message from the parameter operating program, the drawing new element program transforms co-ordinates for the data of the NewBuffer. The co-ordinates written in the NewBuffer were the sum of the element off-set co-ordinates to the current SDB and the current SDB segment co-ordinates. To get the relevant off-set co-ordinates, the segment co-ordinates should be deducted. After this co-ordinates transform, the drawing new element program can now move these co-ordinates to the current SDB buffer as one of the elements that locate on the current SDB. It means that this element has now inherited the index keyword of the current SDB and become a member of this SDB data set. This drawing new element program is illustrated by the flowchart shown in Fig.6.25.

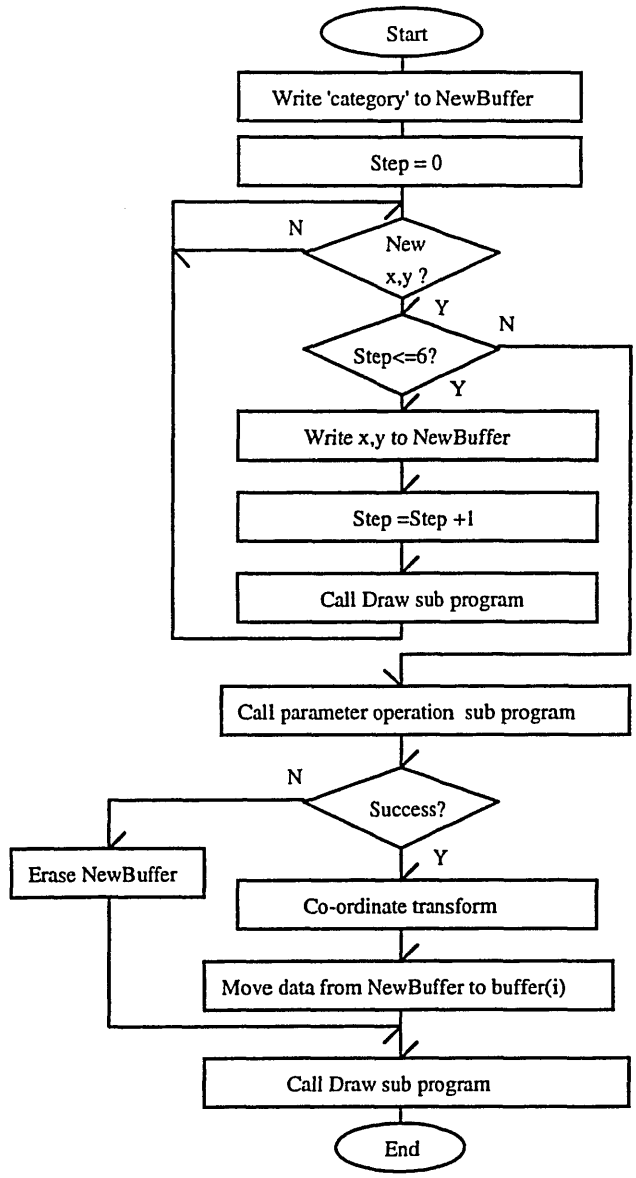


Fig.6.25 Drawing new element program flowchart

6.7.4.2 Low level sub programs

The low level sub programs discussed here include program 27: moving step counter, 28: move buffer data, 29: load new SDB data and 26: draw.

27	moving step counter

The moving step counter program is the common sub program for Move drawing board left/ right/up/down programs. This program updates the Moving status table that

records the drawing board moving operation series according to the keywords MAK and LK inherited from the drawing board moving program.

In the Drawing Diagram program, diagrams on High(H), Middle(M) and Low(L) layers of the drawing board are displayed in the proportion of 1:3:9. When two or three layers are in an overlapped working status, the *segment co-ordinate moving* (see the definition given in program 6: Moving drawing board left) of each upper layer is based on that *moving* of its adjacent lower layer. For example, in the circumstances where the H and M layers are both active, each time the M layer sub drawing boards have their three step left (right, up, down) *segment co-ordinate moving*, the H layer sub drawing boards will have one step left (right, up, down) *segment co-ordinate moving*. To record these movements, the Moving status table is designed as shown in table 6.1, where U , D, R and Lf stand for up, down, right and left respectively.

Layer	MoveKey				Step-N			
	U	D	R	Lf	U	D	R	Lf
H	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0

Table 6.1 Initialised Moving status table

In this table, H, M and L is to identify the layer in which the movement happened. The value of MoveKey (U/D/R/Lf) can be either 1 or 0, and 1/0 stands for there has been/was no *segment co-ordinate moving* requirement to the direction of U/D/R/Lf. The value of Step-N (U/D/ R/Lf) could be any positive integer including 0, which stands for the historically cumulated number of the moved steps.

The MoveKey part of the Moving status table is used by the Move buffer data and Load new sub drawing board data programs (detailed later) to obtain moving

information. Once the information has been obtained by these two programs, all the records of the MoveKey part will be erased. The Step-N part is for the Draw program to obtain the new diagram location. The record in this part will be kept during a complete Power Shell operation process. The use of the Moving status table and the working process of the Moving step counter program can be further explained by the following illustrative example.

Suppose H and M are the currently active layers, the user has operated right moving twice and then left moving once on the M layer. In this moving process the Moving status table has been kept updated, which is illustrated in Fig.6.26. The result in the Step-N part of the last table of Fig.6.26 is synthesised from this three- step operation, i.e. 'right 2 + left 1= right 1'. In the above example the moving status table has provided the following information for the program 28: Move buffer data and program 29: Load new sub drawing board data:

- M layer needs to be moved (all three times);
- the moving direction is right (the first move), right (the second move) and left (the third move).

In addition, it has also provided a moving measure for the program 26: Draw. For example, the last table of Fig. 6.26, the data from the Step-N part means M layer should move one step right, therefore the H layer should move right 1/3 moving step, i.e. the H layer segment co-ordinates is required to be translated to 1/3 unit.

Because the proportion of the H, M and L layer is 1:3:9, the ternary notation has been adopted in the moving step counter. For example, when the M layer has moved to the right twice, the third time save operation will cause a carry-over, i.e. in the Step-N part the value for H-R will be 1, and for M-R will be 0. Meanwhile the H-R value in MoveKey part will also change to 1 to indicate the movement required of the H layer.

From the initial table given in Table 6.1

after receipt of the first 'right moving' message,
⇓ updated by Moving step counter

Layer	MoveKey				Step-N			
	U	D	R	Lf	U	D	R	Lf
H	0	0	0	0	0	0	0	0
M	0	0	1	0	0	0	1	0
L	0	0	0	0	0	0	0	0

read by program 26, 28 and 29(detailed later),
⇓ updated by program 27: Load new drawing board data

Layer	MoveKey				Step-N			
	U	D	R	Lf	U	D	R	Lf
H	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	1	0
L	0	0	0	0	0	0	0	0

after receipt of the second 'right moving' message,
⇓ updated by Moving step counter

Layer	MoveKey				Step-N			
	U	D	R	Lf	U	D	R	Lf
H	0	0	0	0	0	0	0	0
M	0	0	1	0	0	0	2	0
L	0	0	0	0	0	0	0	0

read by program 26, 28 and 29(detailed later),
⇓ updated by program 27: Load new drawing board data

Layer	MoveKey				Step-N			
	U	D	R	Lf	U	D	R	Lf
H	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	2	0
L	0	0	0	0	0	0	0	0

after receipt of the 'left moving' message,
⇓ updated by Moving step counter

Layer	MoveKey				Step-N			
	U	D	R	Lf	U	D	R	Lf
H	0	0	0	0	0	0	0	0
M	0	0	0	1	0	0	1	0
L	0	0	0	0	0	0	0	0

Fig.6.26 Moving status table updating process

28	move buffer data

The move buffer data program is the common sub program for Move drawing board left/ right/up/down programs. This program reads the Moving status table that records

the drawing board moving operation series according to the keyword LK inherited from the drawing board moving program. The reading will start from the layer indicated by LK to its upper layer(s). The information searched includes:

- whether any *segment co-ordinate moving* is required;
- the direction of movement.

When moving is required the move buffer data program will move the diagram drawing data in the buffer according to the information obtained. The program flowchart is given in Fig. 6.27.

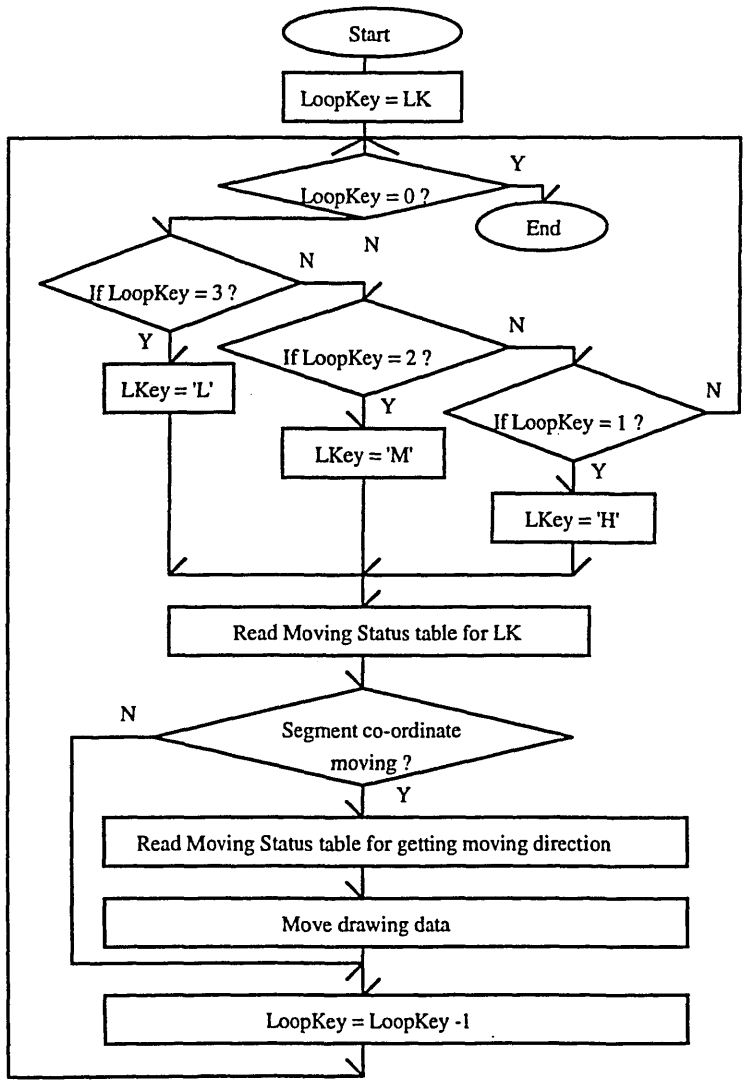


Fig. 6.27 Move buffer data program flowchart

29	Load new sub drawing board data

The Load new sub drawing board data program is the common sub program for Move drawing board left/ right/up/down programs. This program requires information from the Moving status table and works on the projection matrices for H, M and L layer. From the Moving status table, it obtains the sub drawing board that has been required for a segment co-ordinate moving and the moving direction.

It then, according to the direction of movement, identifies the line(s) or columns of the projection matrice(s), the elements of which should be updated, i.e. load new data to the corresponding buffers. For example, a left movement is required, the buffer corresponding to elements on the first right column of the projection matrix should be loaded in the new data, as shown in Fig.6.28.

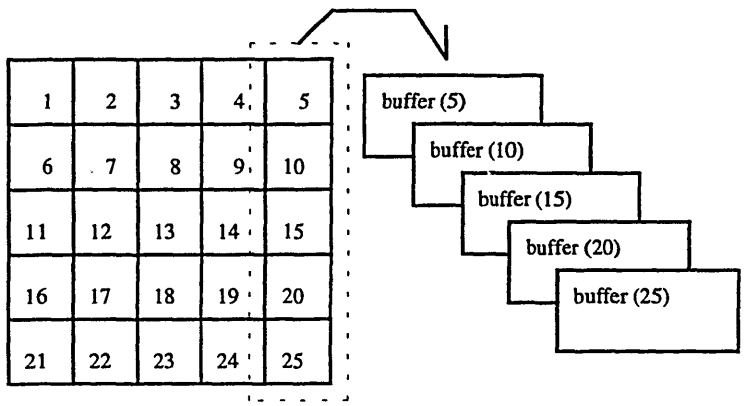


Fig. 6.28 Load in new data to the corresponding buffer

Next, using the duplex pointer(s) of the projection matrice(s) the load data program produces the index keywords for the sub drawing board buffers that should be loaded with new data. Finally, the database drawing data table should be opened and according to the index keyword(s) data loaded in to the sub drawing board buffers. The flowchart for this program is shown in Fig.6.29.

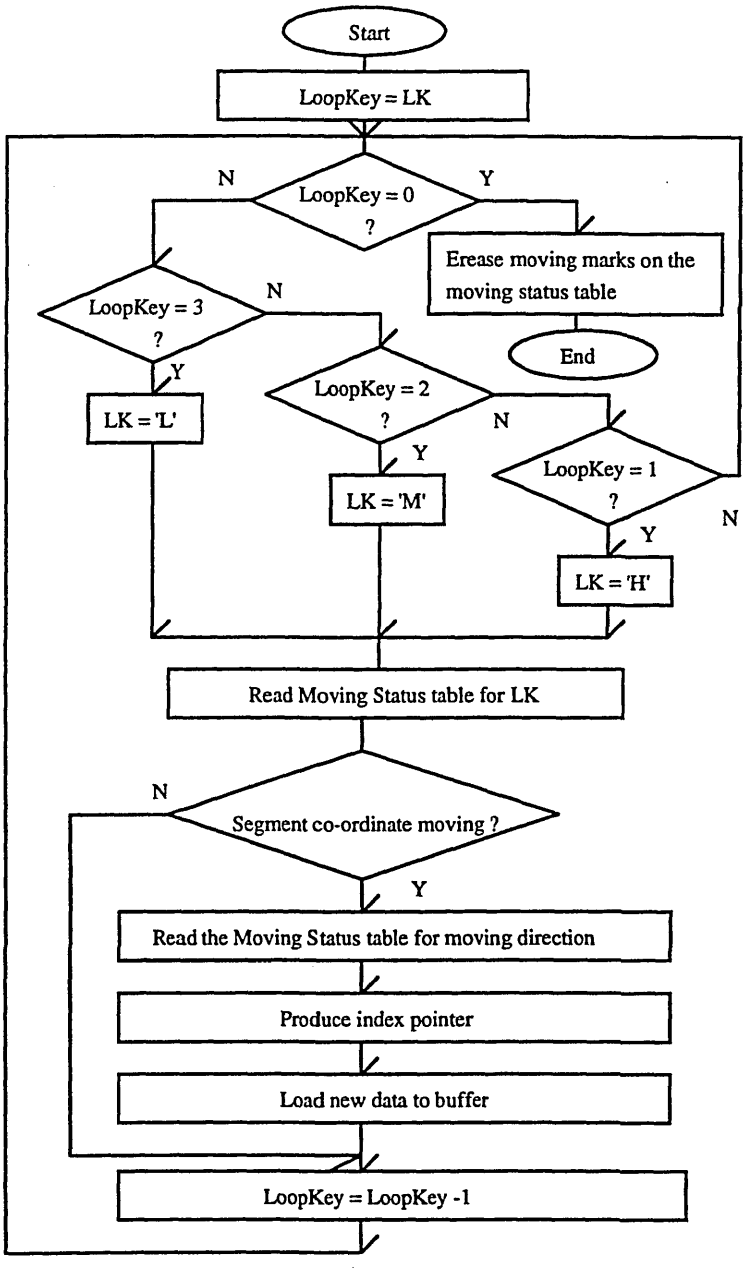


Fig.6.29 Load new sub drawing board data program flowchart

26	draw

The Draw program is a lower level sub program of the Drawing Diagram program. It will be called by several operations as the final step for refreshing windows and drawing diagrams. These operations include:

- moving drawing board left, right, up and down;
- zoom in/out drawing board;

- drawing or updating power system element symbols.

There are six drawing symbol functions in this Draw program. These are:

- Drawing Busbar(),
- Drawing Cable(),
- Drawing Transformer(),
- Drawing Load(),
- Drawing Generator(), and
- Drawing Map().

To call the Draw program, a drawing task table has to be filled by the calling operations. The contents of the table include:

(1) Sub Drawing Board (SDB) buffer(i), where

$i = 1, 2, ..., 25$, for H layer;

$i = 26, 27, ..., 26+24$, for M layer; and

$i = 51, 52, ..., 51+24$, for L layer.

Each SDB buffer(i) can load in nine drawing data records. Each of these records contains the following fields:

- power system element name;
- element category, i.e. Busbar(B), Cable(C), Transformer(T), Load(L), Generator(G) and Map(M);
- element breaker status;
- element off-set co-ordinates on the SDB;

  .....

The drawing data in the SDB buffer is either derived from the Drawing Data Table via the Load new SDB buffer data program, or input as new element drawing data by the user.

(2) SDB buffer(i) segment co-ordinates

The SDB buffer(i) segment co-ordinates consist of a constant set that contains 25 elements. It is structured as $\begin{cases} segment[i].x \\ segment[i].y \end{cases}$, $i = 1, 2, ..., 25$.

It is used for the Draw program to decide the drawing location of the drawing data in the SDB buffer[i].

(3) Moving status table

(4) Current drawing layer, i.e. LK inherited from the moving drawing board program.

The Moving status table and LK are used for the Draw program to calculate the segment co-ordinates transform parameter in the moving drawing board that involves more than one drawing layer.

The flowchart of the Draw program is given in Fig.6.30.

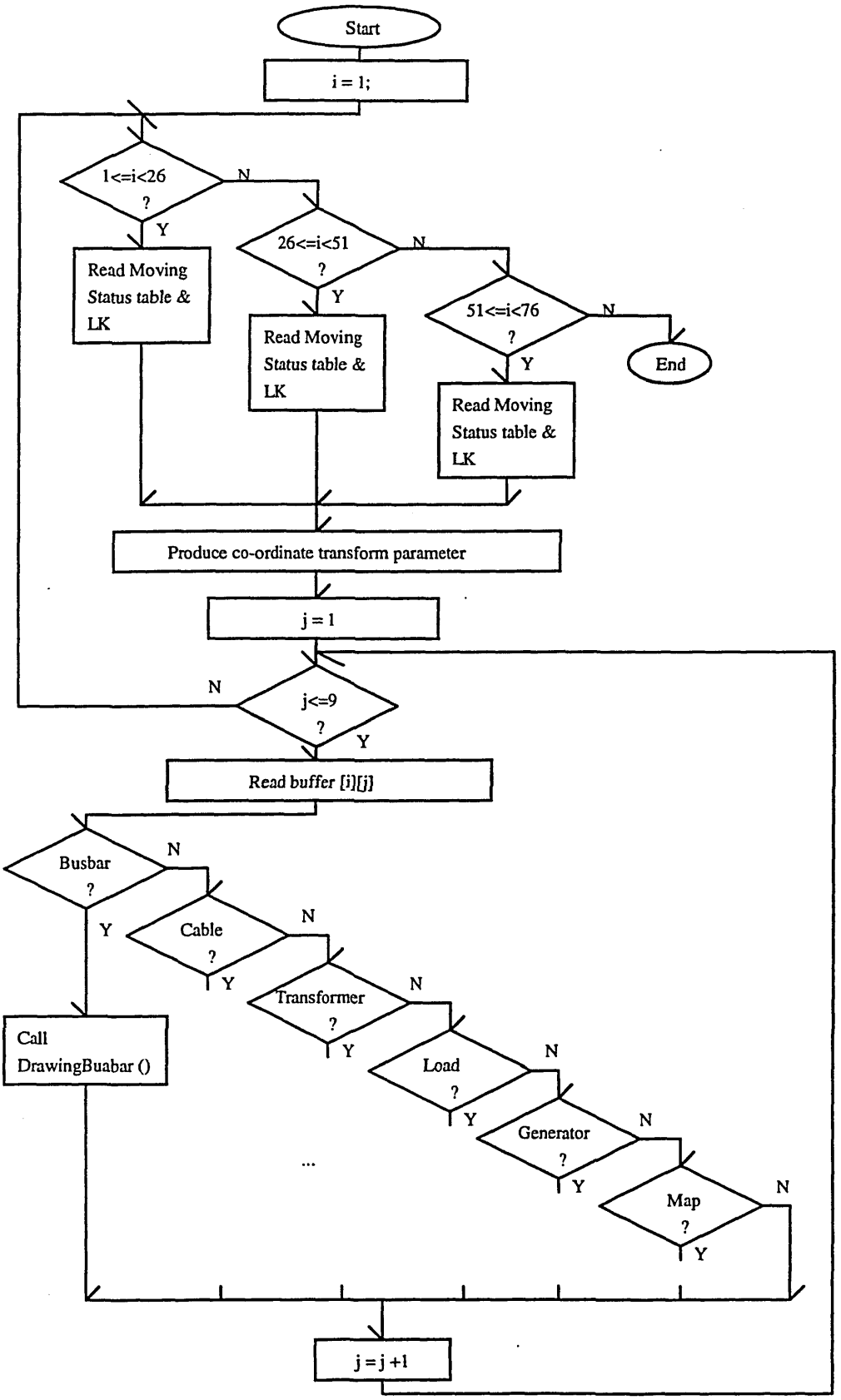


Fig.6.30 Draw program flowchart

6.7.5 Redefine Parameter

As mentioned previously, the Power Shell system has been designed to accommodate different data formats for various power system application programs. This has been achieved by allowing users to redefine data table structures through the Redefine Parameter program. The parameter redefinition window is shown in Fig.6.31.

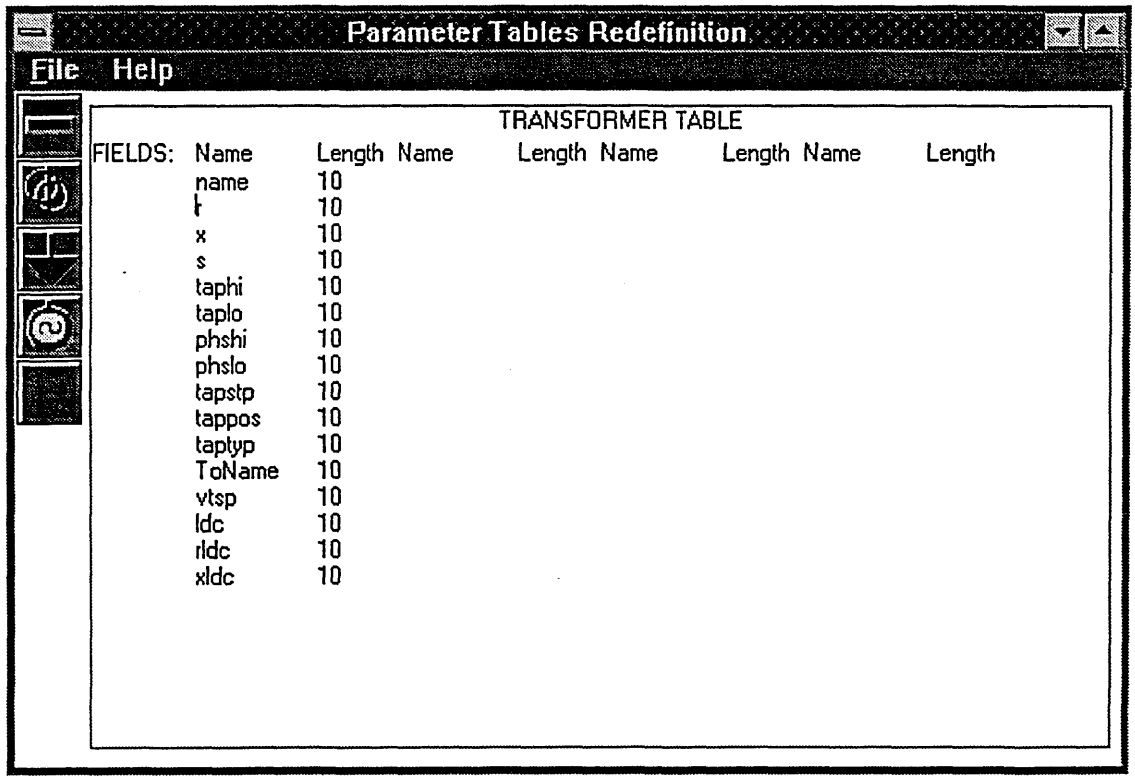


Fig.6.31 Parameter redefinition window

When a Power System Project is created through the Create Project program, seven database tables are produced on the hard disk. These seven tables are used to record descriptional data for different aspects of the power system network, which include five element-parameter tables:

- Busbar table,
- Load table,
- Generator table,
- Cable(Line) table,

- Transformer table.

A default definition of the name and the length of each field in the table is given by the Create Project program according to a normal routine. If a different structure is required, through the Redefine Parameter program, the user can redefine the name and length for these tables. This includes adding or deleting the fields in the table and changing names and lengths of the fields. After the redefinition procedure, new data tables based on the structure redefined will be produced by the Redefine Parameter. In order not to break the relations between these tables, the first field of each table, the element name, used as the index keyword for the Power Shell system, is not allowed to be redefined.

The Redefine Parameter program has been implemented by using relevant database techniques. The data tables in the database are actually the disk files allowing hybrid data types. The head part of each file is a table that describes the organisation structure of this file. The head table indicates the total number of fields of each record in the file, field name, data type, field length and so on. This head table is called the database data dictionary. The data structure of a database data table will be renewed by the updating of the data dictionary. To construct and produce the data dictionary in conformation with industry standards, the corresponding data table will be compatible with other commercial database systems. The Redefine Parameter program is illustrated by the flowchart shown in Fig.6.32.

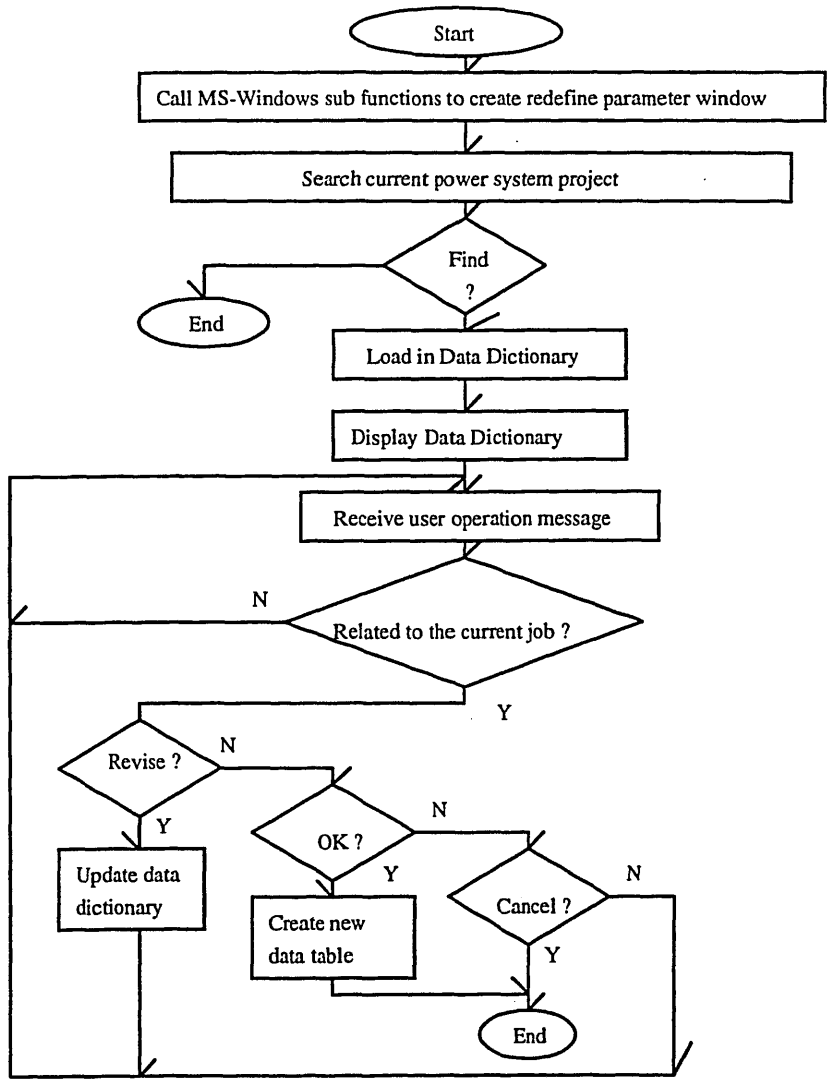


Fig.6.32 Redefine Parameter program flowchart

6.7.6 Copy Project

Drawing one-line diagrams of a power system network is very time consuming. Sometimes a new Power System Project is created just because a different data format has been required for a new application program rather than a new Project for a different power system network. In this case the diagram, and probably also some data for the new Project could be identical with that of an existing Power System Project. Thus copying of the identical part to the new Project will save effort in creating a new

Project, especially drawing a one-line diagram. The Copy Project program is to provide this facility to the Power Shell users.

The Copy Project program is also based on data dictionary techniques. In the operation, the program opens both source and target data tables, and copies the data of the source table to the same field of the target table according to the data dictionaries of these two tables. If a field of the source table does not have its identical field in the target table, then the data of this field will be ignored during the copying process.

The Copy Project program has provided users with an object oriented operation style. After users designate the source and target Project, the seven data tables of the source Project will be automatically copied to the target Project.

6.7.7 Data Export

The Power Shell system has been designed as the LIME of the DIMS. It does not contain any power system application programs, but is required to provide source data files for the application programs to read in for calculations. The current situation is that these power system application programs have been built with individual data formats which are not at all identical to each other. This individuality can be summarised as follows:

(1) for the same category elements, different applications require different parameters or the same parameters but in a different order;

(2) different applications require different prefix styles for data records. The prefix of a record is normally for describing the relationships of the power system elements. So far there is no international standards for prefix styles for power system applications. Therefore different companies have applied their own prefix styles in the implementation of these power system application packages. An example of different prefix styles used for Cable records by ABB and PSS/E as follows:

ABB:

FromBusbarName, ToBusbarName, Parameter(1), Parameter(2), ...;

PSS/E:

FromBusbarNumber, ToBusbarNumber, CableNumber, Parameter(a), Parameter(b), ...;

Dealing with data format difference, therefore, has become the key issue in enabling the Power Shell system to serve various application programs. This has been achieved, first, as mentioned in section 6.7.5, by a Redefine Parameter facility which allows users to re-organise parameter formats of Power Shell files to match their application program formats. It has then been supplemented by the Data Export program that relates the redefined parameter formats to user-required prefix styles as complete records and produces ASCII files in this completed structure as the input data files required for the power system application programs.

The major tasks of the Data Export program include:

- support users to select prefix styles, the selection window is shown in Fig.6.33.
- set relations between the network topology table and the parameter tables according to user requirements, i.e. add the element connection descriptions given by the Network topology table as the prefix to the parameters in the parameter table. This is for producing a complete record, which consists of prefix and parameters, for power system application programs.
- write parameters with prefix into ASCII files in the order given by the database table data structures.

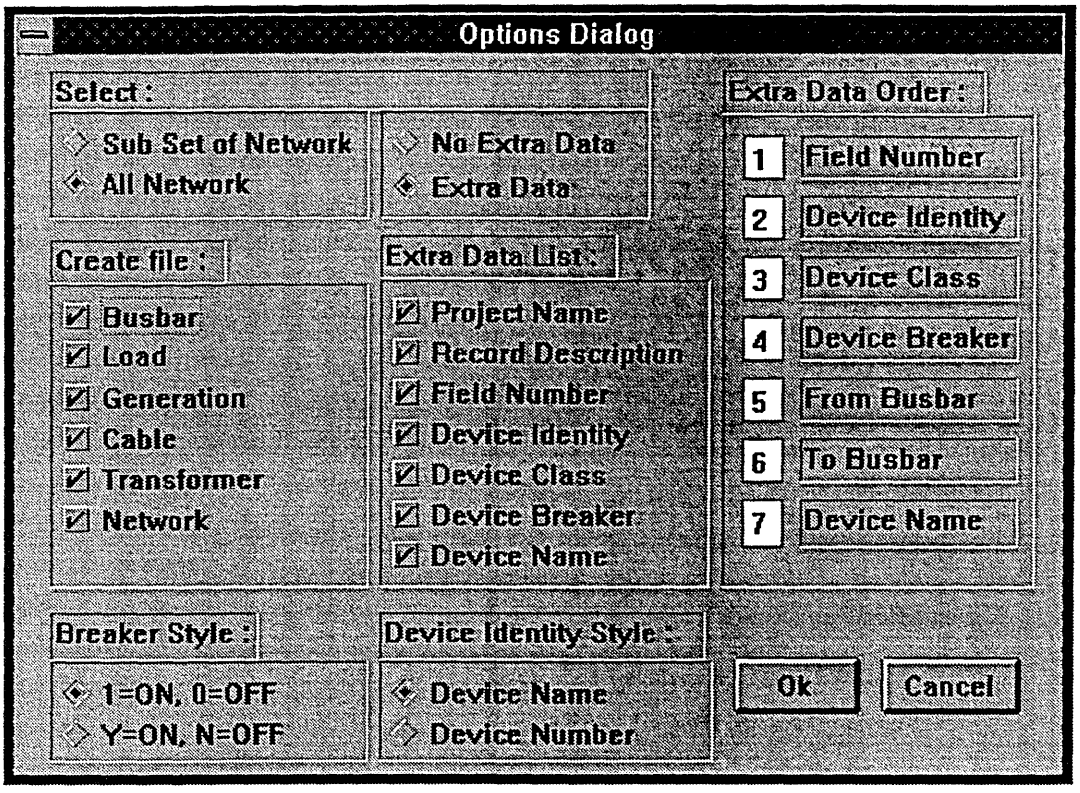


Fig.6.33 Data record prefix style selection window

The Data Export process is described by the program flowchart given in Fig.6.34.

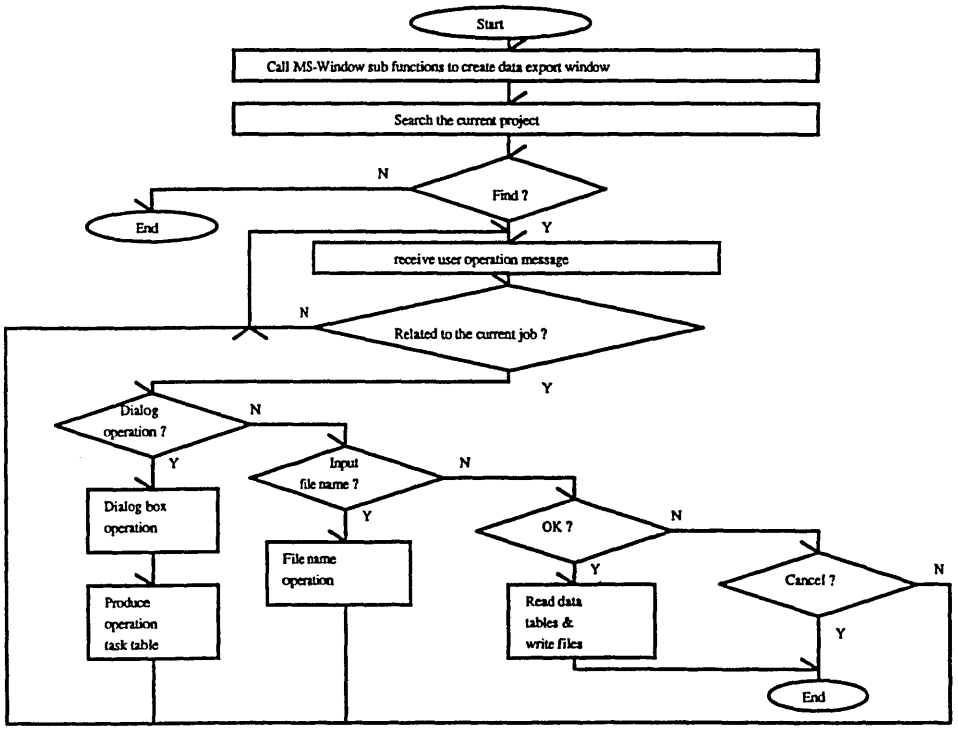


Fig.6.34 Data Export program flowchart

6.7.8 Data Read Back

Drawing one-line diagrams for power system applications has provided benefits for viewing power system network topologies. Further facilities for displaying the network element running status data or application program calculation results on the screen diagram will enable the user to be informed visually regarding the network working situation. Thus the Data Read Back program has been employed for providing such facilities. Through the Data Read Back program, the data in an ASCII file in the format specified by the Power Shell system can be read back to the Power Shell database tables and then be displayed on the screen diagram.

The ASCII file format designated by the Power Shell system for reading data back consists of the following ordered fields:

Element category, Element name, Information class and Element status key, Information;

where

Element category: for specifying the element category, i.e. Busbar(B), Cable(C), Transformer (T), Load(L) and Generator(G);

Information class: for specifying what the information is about, such as Voltage and Current;

Element status key: for indicating device running status, and it could be any one of '=', 'L(or 'l'), and 'O'(or 'o') standing for Normal, Low and Over respectively.

For example, the data records of an ASCII file to be read back could be:

C C1 V= 300;

C C2 VL 285;

C C8 Vo 320;

where C stands for the element category Cable; C1, C2 and C3 are the names of the cables; V stands for Voltage; '=', 'L' and 'o' are the running status of C1, C2 and C3 respectively; the last data column is the voltage rating of the corresponding cables.

6.7.9 Parameter Table Operation

The Parameter Table Operation(PTO) program cannot be driven via menus since it is for providing service to the Drawing Diagram(DD) program. It will be in a working status only if the DD program is called. When the user inputs a new element or queries a existing element in the DD operating environment, the DD program will send a message to execute the PTO program via the MS-Windows. Then the PTO program will run in parallel with the DD program. These two programs communicate with each other via Data Dynamic Exchange(DDE). The purposes of the communication include:

- DD informs PTO regarding the service required;
- DD informs PTO regarding the data processing target;
- PTO informs DD if the task has been fulfilled; and
- PTO send back relevant results of data processing.

The information has been exchanged through character strings in a predefined format. The format of the string for sending information from the DD to the PTO is:
Job Key, Element Category, Element Name, Breaker Status, Busbar 1, Busbar 2;
where Job Key could be either 'A' or 'U', which stands for 'Appending new data to the table' or 'Update or query data of the table'; Element Category could be any one of 'B', 'C', 'T', 'L', 'G' and 'M' standing for Busbar, Cable, Load, Generator and Map respectively. Values of other items depend upon the Job Key.

If the value of Job Key is 'A', the PTO will open the parameter table for the indicated Element Category, the values of the last four items being:

Element Name: empty;

Breaker Status: empty;

Busbar 1: { empty, if Element Category is 'B' or 'M';
input busbar name, if Element Category is 'C' or 'T';
input / output busbar name, if Element Category is 'G' or 'L'.

Busbar 2: { empty, if Element Category is 'B', 'G', 'L', or 'M';
output busbar name, if Element Category is 'C' or 'T'.

If the value of Job Key is 'U', the PTO will open the parameter table for the indicated Element Category, the values of the last four items being:

Element Name: the name of the existing element that is to be updated or being queried and will be used by PTO as the index keyword to find out the corresponding record in the opened parameter table;

Breaker Status: { 1, if the breaker is switched on
0, if the breaker is switched off ;

Busbar 1: empty;

Busbar 2: empty.

The format of the character string for sending information from the PTO to the DD is as follows:

Job Result Key, Element Category, Element Name, Breaker Status, Busbar 1, Busbar 2;

where the only difference from the first string is the Job Result Key.

Job Result Key is for specifying user operations. It's value could be one of 'O', 'C', or 'D' that stand for OK, Cancel and Delete respectively. After inputting, or updating, or querying any data in the database table, the user can choose OK or Cancel or Delete to

make the operation become effective. For each of the selections the PTO program will make the following corresponding responses:

OK: write the data, which has been put in the PTO window, to the database table, and then report the result of OK to the DD program.

Cancel: ignore the data in the PTO window, and then report the result of Cancel to the DD program.

Delete: according to the index keyword of the current record in the PTO window, delete the corresponding record in the database table, and then report the result of Delete to the DD program.

The report sent by the PTO program will effect the DD program executing corresponding sub programs.

(1) After the DD 'Appending' job has been achieved by the PTO program

- In the PTO job result report, the Job Result Key is 'O', i.e. OK

The DD program will write the drawing data of the new appended element, together with the Element Name and Breaker Status sent by the PTO, to the database Drawing data table. It will also write the Element Name, Busbar 1 and Busbar 2 to the database Network topology table for identifying the relationships of the new element within the power system network.

- In the PTO job result report, the Job Result Key is 'C' or 'D', i.e. Cancel or Delete

The DD program will ignore and delete the data of the involved element.

(2) After the DD 'Update' job has been achieved by the PTO program

- In the PTO job result report, the Job Result Key is 'O', i.e. OK

The DD program will update the data of the identified element and write renewed data to the database Drawing data table and Network topology table as above.

- In the PTO job result report, the Job Result Key is 'C', i.e. Cancel

The DD program will consider the user operation as a 'read only' process, and will not update any data.

- In the PTO job result report, the Job Result Key is 'D', i.e. Delete

The DD program will delete the record of the currently identified element from the Drawing data table and Network topology table.

Furthermore, to support the user to redefine parameter table structures, the PTO program must be able to automatically update its window format in meeting the requirement of the redefinition, and this has to be done before starting any of the above services for the DD program. Here, again, the database data dictionary technique has been used. After receiving the element category message from the DD program, the PTO will search the currently active Power System Project in the project table provided by the Project Manager. When the current Project has been found, the PTO program will, according to the category keyword, open the relevant parameter table of the current Project, and then format its operation window to suit the data structure given by the data dictionary of the parameter table. The purpose of this operation window formatting is to assign the display location co-ordinate for each data field in the renewed structure. After formatting the operation window, the PTO will start to serve the DD program.

The PTO program flowchart is given in Fig.6.35

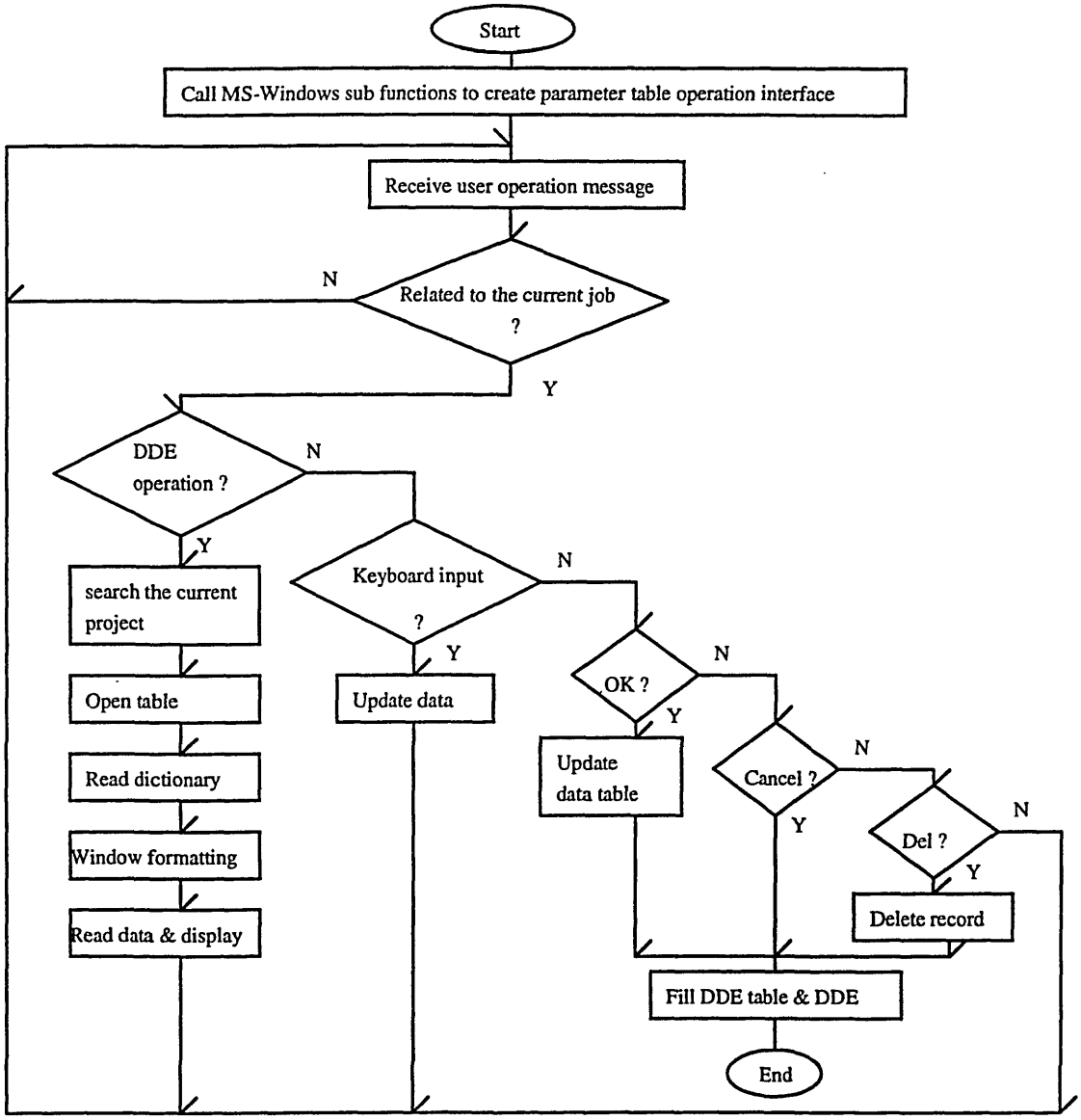


Fig.6.35 Parameter Table Operation program flowchart

6.8 User Guide

In addition to the development work described in the above sections, the Power Shell User's Guide has also been completed in order to render the Power Shell system a practical tool for power system applications. The User's Guide has been written after consulting some commercial software user guides. In the User's Guide the functions and operation procedures of the Power Shell system have been detailed. The Power Shell User's Guide is detailed in the appendix of this thesis.

6.9 Summary

The requirements for the LIME-Power Shell have been specified in the previous chapters. These include managing network one-line diagrams, network topology and element parameters for very large scale power systems, accommodating various power system applications with different data formats, and being compatible with commercial software tools. In order to meet these functional requirements and also for other quality characteristics including system consistence, usability and maintainability, the Power Shell system has been implemented following standard software development methods and using a number of programming techniques. The implementation involves the analysis of the requirements of system input and output, internal data structure, system operating requirements and program running characteristics; the development tool selection; and the design of the program structure and sub job modules. These are the modules for system scheduling, project management, drawing diagram, parameter operating interface, data structure redefinition, project copy and data exchange. The programming techniques that have also been used include software graphical techniques, Windows techniques, database techniques, object oriented operation techniques, data dictionary techniques, modular programming techniques, top-down implementation techniques and multi-job concurrency techniques.

As a running system, Power Shell has been tested using data from industrial power system networks and evaluated against the specified requirements and the design objectives. These are detailed in the next chapter.

CHAPTER 7

EVALUATION OF THE LIME

In this chapter the LIME-Power Shell is evaluated. Power Shell, as the basis of a DIMS, has been developed for a common information management environment for various power system applications rather than the operating environment part of a specific application program. So far no other common environments for power system applications have ever been identified. The concept was, in fact, proposed by this research. Therefore the evaluation has to be made mainly against the requirements and objectives discussed in Chapter 2, on which basis Power Shell has been designed and implemented. These include:

- to enable local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and element parameters for very large scale power systems;
- to provide the user with database data structure redefinition facilities, thereby allowing the accommodation of various power system applications that require different input ASCII file data formats;
- to be compatible with commercial word processing and database systems to benefit users with access to different software tools.

On the other hand, during the period of Power Shell development, the power system application program development techniques were being rapidly updated. In 1992, at the time this research started, there was a discussion on "Interactive graphic power system analysis programs" at the IEE Colloquium(March 92). The discussion involved the user interface of power system application programs, but no requirements on flexible database data structure management had been stressed. Two years later, as the Power Shell development had been completed, at the IEE Colloquium(Nov.94), the discussion was then concentrated on "Developments with interfacing of power system analysis software with SCADA and data management systems". This concerns flexible data structures for different power system applications, the problems of large

data amounts for data management environment, and also data location issue for a distributed information management environment. The issues have been discussed from the viewpoints of both power system application program developers and users. The developers included those who are from leading software companies such as ICL, Ferranti Syseca Ltd, IBM and Power Technologies. The users were mainly from major power system management companies such as National Grid and Hydro-Electric. Therefore their opinions on the power system application requirements and software development requirements could be taken as a professional standard for the confirmation of the Power Shell design objectives.

The evaluation made in this chapter has been focused on the following aspects of Power Shell:

- capability of managing information for large scale power systems;
- database management method;
- display on-line or calculation data on one-line network diagram;
- data exchange with commercial software; and
- accommodation of various application programs.

7.1 Managing information for large scale power systems

The capability of managing information for very large scale power systems is one of the major objectives of the Power Shell design. The need for this has been confirmed by the IEE Colloquium(Nov. 94) discussion concerning the size of Distribution networks and the use of a Central database.

The size of distributed network has been described by Parton [PART94], the consultant to the ICL DINIS program. *The basic size of distribution networks within*

one PLC is substantial, for example, secondary (11KV/415V) substations can number up to 40,000, the 11KV network may have up to 1 million supporting wood poles (all expecting to be uniquely identified), and for network analysis studies, inter-connected networks of several thousand active nodes can be quickly generated. Indeed, solutions of up to 100,000 nodes have been reported. Apart from the size of the network itself, the supporting data needed for detailed planning studies can be equally large. For example, some users are finding the need for over 1,000 different line codes, with all possible combinations of different conductor types in each phase and neutral. ..... Furthermore, when considering digitised map background requirements and other associated (GIS) data, current technology is indicating over 1 Mbyte for each 1:10,000 stored map, so that putting all the information together is generating data storage needs in the multi Giga byte range. As seen from this description, the data amount involved in a professional power system has exceeded the internal memory capacity of current PC platforms. Furthermore the Central database systems has been suggested for recording power system data. It has also been indicated by Parton: "*Thus the practical data design principle becomes to leave the main bulk of specialist data in a simple direct access and updatable form within the Dept responsible, but retain keys to this via a generic central data recording system.*" On the other hand, as discussed in the Chapter 2, "*Most importantly, operators at distribution zone offices and power plant control rooms have access to the same one-line diagrams and tabular displays as the system operators* [TREF88]". These statements have confirmed the necessity of this 'managing large amount data' objective in the Power Shell development, which is for supporting local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and parameters for large scale power systems. This objective has been achieved by using the data managing and scheduling method discussed in Chapter 4, 5, and 6.

The Power Shell capability of managing information for large scale power systems has been tested against the data of a published NGC 400/275 KV transmission network. The network contains 564 elements and is the only large scale PSN for which the data is available. The network topology, elements and one-line network diagram have been input to Power Shell through the drawing one-line diagram operation. All the seven database tables have then been created by Power Shell. During test trials Power Shell has functioned well and performed consistently. No data overflow has occurred. An example of a Cable data table produced by Power Shell is shown in Fig.7.1, and the one-line diagram of this NGC network displayed via Power Shell diagram interface is illustrated in Fig.7.2.

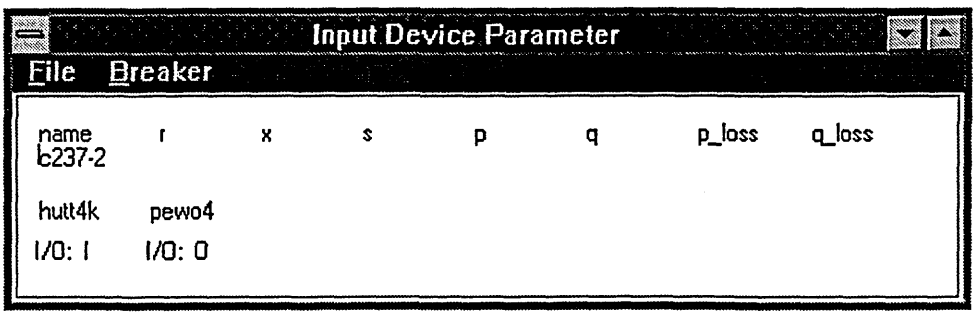


Fig.7.1 An example of the NGC data tables produced by Power Shell

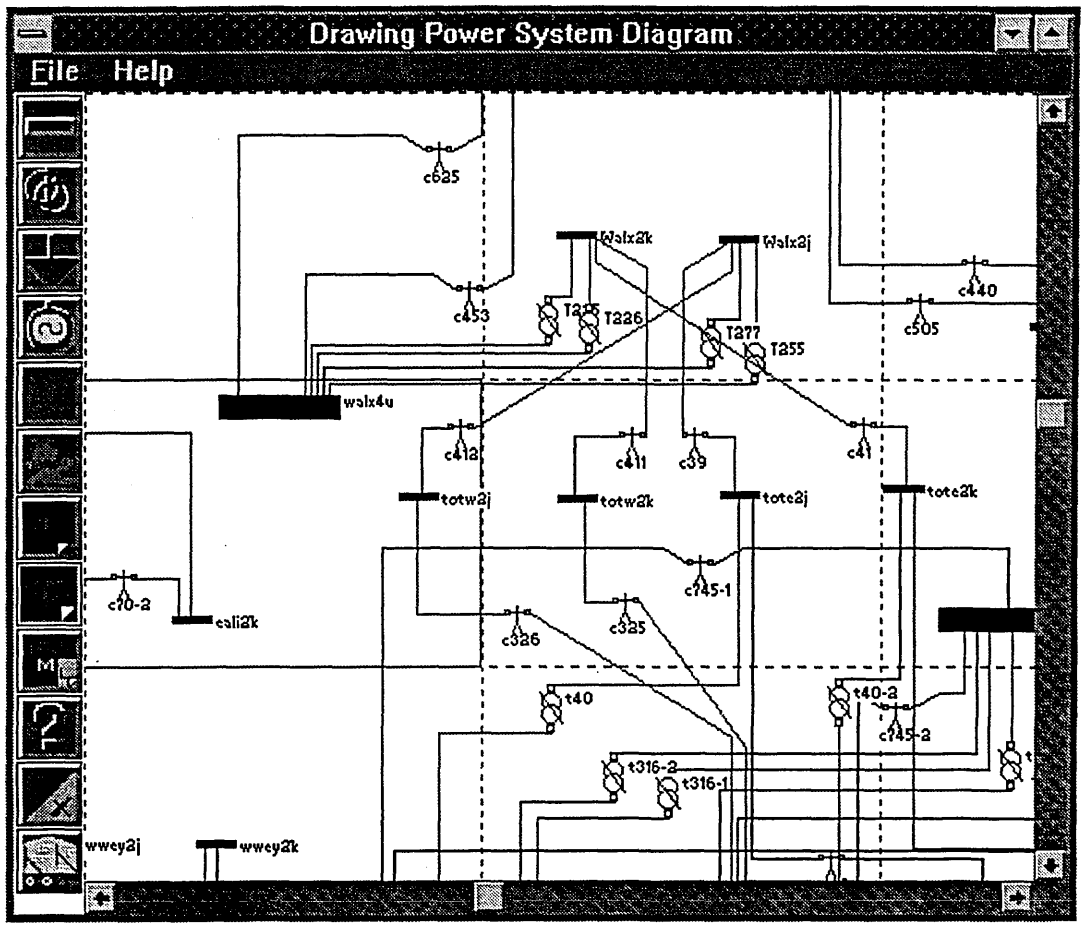


Fig.7.2 NGC network one-line diagram

Therefore Power Shell could provide users with at least the following two major benefits:

- access to power system data and one-line diagrams managed by the Central database of the host computer from local PC platforms via Power Shell;
- enable an effective data communication in the DIMS. Since the network topology, element parameter and one-line diagram can be stored and managed by Power Shell on the local PC platform, when the power system running status information is required, the host computer system need send only the on-line data through LAN.

7.2 Database management method

As mentioned previously, in the design of the database management method for Power Shell, the flexibility of the data structure has been considered as one of the key points. Therefore the relational database data dictionary technique has been used to provide users with the possibility of redefining the database data table structure; the object oriented database management method has been used to support users in the creation and management of well indexed data tables of Power System Projects. The necessity of allowing a flexible database data structure has been increasingly recognised since then. In the IEE Colloquium (Nov.94), Mr Hainsworth from Ferraniti Syseca Ltd described the flexibility as one of the database design features[HAIN94]: "*A feature of the data base design is that it allows the customer significant freedom in specifying the structure of the data base. It also accommodates updates to the table and data base structure without the need for software modification.*" This has also been confirmed, at the same Colloquium, by Mr Parton from ICL in his analysis of the essentials in database design [PART94], which were summarised as: "*a) The first essential is that any data base needs to be extremely flexible in itself, or have a flexible interface with other corporate system. Local data base will constantly change as new requirements and technical options emerge. ..... f) To aid the easy and prompt traffic between different data systems, it would appear attractive to interpose a universal data base system, probably using relational database tools, to act as a general purpose gateway between all other systems. Not only can this ensure a common point of information access, but enables any major new system developments to immediately become part of the data system, and hence enable technical users to extract any data for new technical programs or queries.*"

The achievement of the design objective, i.e. allowing flexible database data structure, has enabled users to build individual Power System Projects and update data formats when necessary without breaking down the data relationships. This forms the basis of the success of the accommodation of different application programs as detailed in section 7.5. Therefore the suitability of the techniques and method used in meeting the objective has been demonstrated.

7.3 Display on-line or calculation data on one-line diagram

The necessity and importance of reflecting on-line data or calculation result via one-line diagram has been stressed at the recent IEE Colloquium (Nov.94) for Power Division [PART94] [HAIN94]. This therefore has confirmed the design objective of Power Shell in providing such functions.

In system operation, the function has been achieved by first reading the relevant ASCII files back to the database table and then displaying data on the one-line diagram. Also the power system element current status is described by the display styles. A test of this function has been made by using an illustrative example. The following is the example ASCII file for the test, and the display of the data on the corresponding one-line diagram is shown in Fig.7.3.

Example text file:

```
95
B SS19 V= 300
B SS23 VL 299.8
B SS24 Vl 288.001
B SS25 V= 299.92
B SS26 V= 300
B SS18 V= 299.99
B SS15 v= 300
B SS16 V= 300.2
B SS20 V= 300
B SS21 V= 299.80
B SS22 V= 300.01
B SS29 Vo 300.5
B SS30 VO 300.8
B SS12 V= 300
B SS14 VO 301
B SS17 V= 300.001
B SS10 VL 280
B SS27 V= 300
B SS13 V= 175.03
B SS11 V= 174.09
B SS28 VL 173.0
B SS9 VO 176
B SS1 Vo 176
B SS2 V= 175.00
B SS3 V= 174.99
B SS4 V= 174.98
B SS6 V= 175.0
B SS8 V= 174.09
B SS5 V= 175.000
B SS7 VL 173.492
C TL23 X= 0.05750
C TL30 X= 0.18520
C TL23 X= 0.17370
C TL33 X= 0.03790
C TL34 X= 0.19830
C TL20 X= 0.17630
C TL21 X= 0.04140
C TL22 X= 0.11620
C TL24 X= 0.08200
C TL29 X= 0.04200
C TL31 X= 0.02080
C TL32 XO 0.02180
C TL35 X= 0.55600
C TL37 X= 0.20800
C TL38 X= 0.11000
C TL39 X= 0.25600
C TL17 X= 0.14000
C TL18 X= 0.05750
C TL19 X= 0.18520
C TL27 X= 0.17370
C TL25 X= 0.03900
C TL26 X= 0.01983
```

C TL28 X= 0.17630
C TL40 X= 0.04140
C TL1 X= 0.11600
C TL2 X= 0.08200
C TL3 X= 0.04200
C TL4 X= 0.20800
C TL6 X= 0.55600
C TL7 X= 0.20800
C TL9 X= 0.11000
C TL10 X= 0.2560
C TL41 X= 0.1400
C TL5 X= 0.05750
C TL8 X= 0.18520
T TL16 P= 300
T TL5 P= 400
T TL12 PL 500
T TL14 Po 320
T TL13 P= 330
T TL36 P= 220
T TL11 P= 130
L LD16 D= 0.02
L LD19 D= 0.06
L LD20 D= 0.00
L LD21 D= 0.23
L LD12 D= 0.00
L LD15 D= 0.06
L LD13 D= 0.11
L LD14 D= 0.06
L LD17 D= 0.08
L LD18 D= 0.03
L LD22 D= 0.09
L LD23 D= 0.03
L LD11 D= 0.09
L LD1 D= 0.02
L LD2 D= 0.17
L LD3 D= 0.02
L LD8 D= 0.06
L LD5 D= 0.00
L LD6 D= 0.23
L LD7 D= 0.09
G GEN6 D= 0.180
G GEN4 D= 0.540
G GEN5 D= 0.300
G GEN2 D= 0.00
G GEN3 D= 0.03

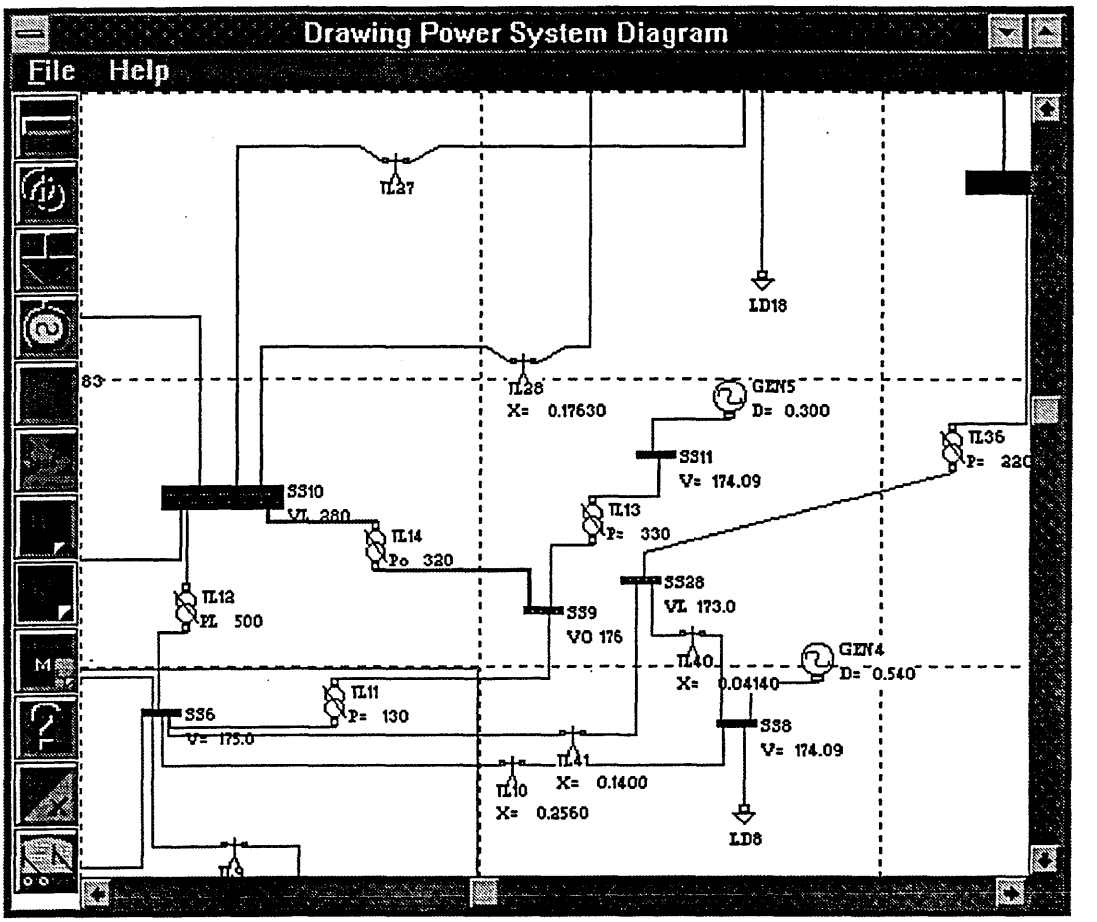


Fig 7.3 The result of displaying relevant data on the one-line diagram

Therefore the function discussed has been demonstrated. The limitation of the current version is that the resulting ASCII file to be read by Power Shell has been restricted to a predefined format. This could be improved to allow users to define the data formats according to their requirements. The improvement can possibly be made by, again, using database data dictionary techniques.

7.4 Data exchange with commercial software

To be compatible with commercial word processing and database systems is one of the objectives of the Power Shell design. This is to enable users to combine the use of different commercial tools. The necessity of having this facility has been confirmed at the IEE Colloquium(Nov. 94). It was indicated by Mr Rawlins from NGC, "*The interfacing of power system analysis software with SCADA data and data management systems cannot be viewed in isolation nor as a technical issue alone. The process must be part of a co-ordinated approach to the use of data within the utility in order to meet its agreed business objectives and in response to both the commercial and technical environment*" [RAWL94].

The above objective has been achieved by using standard MS-Windows techniques and relational database techniques in implementing the Power Shell. The one-line diagram produced by Power Shell can, therefore, be copied to existing commercial software tools such as Microsoft Word and Microsoft Write to generate daily reports. This can be seen by the one-line diagrams shown in this thesis, which have been copied from Power Shell via Microsoft Word. Also the Power Shell database tables can be accessed from other relational database environments such as PARADOX and dBASE IV. This has been demonstrated by a test showing how to access Power Shell database tables from the PARADOX database system. In the test all the relevant Power Shell data tables were easily accessed in the PARADOX environment and an example of a Busbar table is illustrated in Fig.7.4.

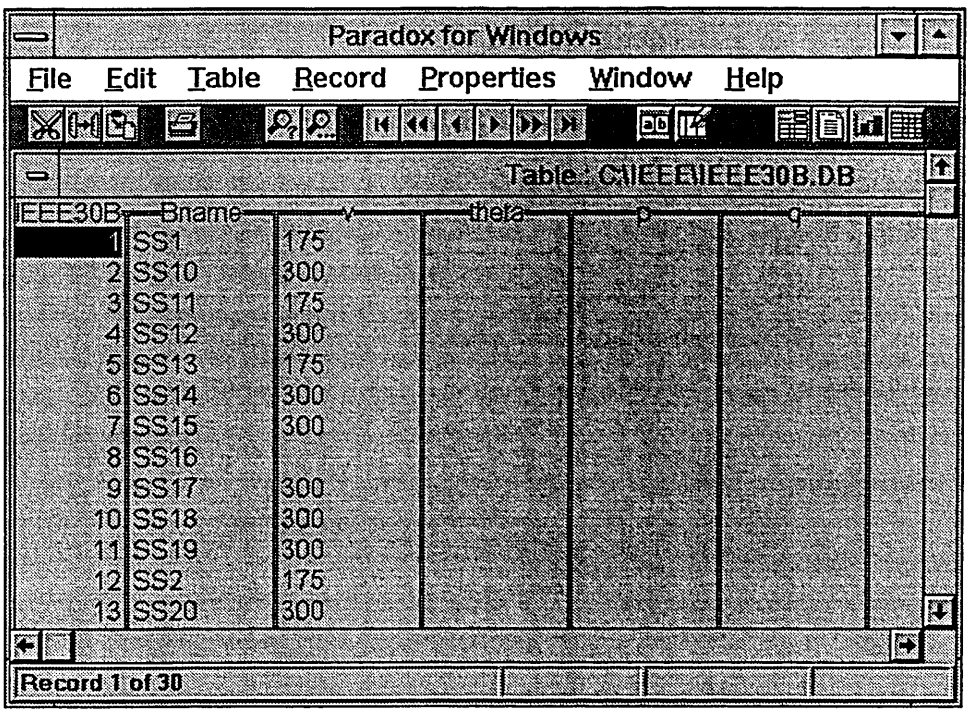


Fig.7.4 Accessing a Busbar table of Power Shell from PARADOX

This has allowed the use of the tools provided by commercial database systems for applying analysis, ordering, indexing, tabulating or statistics calculation to the data in the Power Shell database data tables.

7.5 Accommodation of various application programs

Accommodation of various power system application programs in Power Shell has been achieved by providing users with database structure redefinition facilities supported by the data dictionary techniques and object oriented management method, evaluated earlier in section 7.3. In this section the evaluation focuses on the accommodation capability itself, i.e. the capability of accommodating different data formats used by these applications.

There are four modules employed in Power Shell for information management for power system application programs. These are Create Project, Redefine Parameter, Copy Project, and Data Export. Through these modules a number of different power system applications, which could have different data formats, can be managed in Power Shell as individual Power System Projects. A test to illustrate this has been based on an example application case that involves two power application programs having different data formats: one takes PSS/E format, the other takes Power System Tool Box(Tool Box) format.

Consider a three-person working group. Person A is the power system information operator and is responsible for collecting, organising and updating power system data. Person B is a data analyst and analyses the power system data using PSS/E calculation software package. Person C is also a data analyst but is using Tool Box for data processing. Therefore, besides managing the original data, the operator needs to provide B and C input ASCII files in different formats, i.e. that of PSS/E and Tool Box respectively. The operator's working procedure via Power Shell could be:

1) Gathering the data fields required by either PSS/E or Tool Box for an integrated data set;
2) Create an integrated Power System Project;
3) Redefine data structures for integrated Power System Project element Parameter Tables based on the integrated data sets;
4) Draw the power system one-line diagram and input element parameters;
5) Create a Power System Project for PSS/E application (PSPPSS);
6) Redefine a data structure for power system element Parameter Tables according to the data structure required for PSS/E application. This is, actually, a subset of the initial data structure;
7) Create a Power System Project for Tool Box application(PSPTB);

8) Redefine data structure for power system element Parameter Tables according to the data structure required for Tool Box application;

9) Copy the one-line diagram with the relevant parameters of the integrated Power System Project to the PSPPSS and PSPTB respectively. The Copy Project module of Power Shell will copy only the parameters that are required by the target PSP data formats. After this step the operator will obtain individual data sets for PSS/E and Tool Box respectively.

10) Persons B and C can use the Data Export facilities provided by Power Shell to produce ASCII files as the input data files for PSS/E and Tool Box applications.

If at any time the power system data needs to be updated, the operator would first update the integrated Power System Project, and then repeat step (9), the copy step. Thus B and C will obtain updated ASCII files.

The following is the detailed processing procedure followed by Power Shell for the above example.

1) Gathering data fields from PSS/E and Tool Box formats for an integrated data set

PSS/E Load Flow Activity Descriptions

Bus Data:

I, IDE,PL,QL, GL, BL, IA, VM, VA, NAME, BASKV, ZONE.

Generator Data:

I, ID, PG, QG, QT, QB, VS, IREG, MBASE, ZR, ZX, RT, XT, GTAP, STAT, RMPCT, PT, PB.

Branch Data:

I, J, CKT, R, X, B, RATEA, RATEB, RATEC, RATIO, ANGLE, GI, BI, GJ, BJ, ST.

Transformer Adjustment Data:

I, J, CKT, ICONT, RMA, RMI, VMA, VMI, STEP, TABLE, CNTRL.

Power System Tool Box load flow data structure

Bus Data:

1. B_N = Bus Number
2. V = Voltage
3. A = Angle
4. P_G = P Generation
5. Q_G = Q Generation
6. P_L = P Load
7. Q_L = Q Load
8. G_S = G Shunt
9. B_S = B Shunt
10. B_T = Bus Type

Line Data:

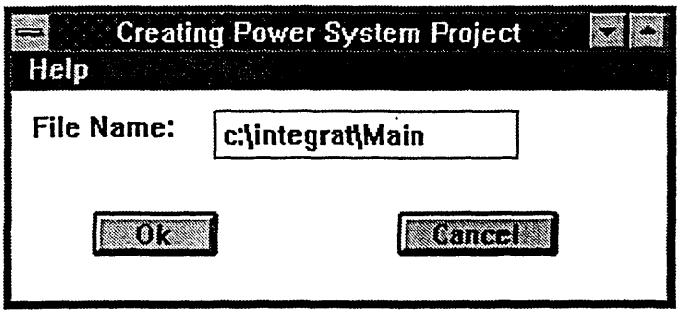
1. F_B = From Bus
2. T_B = To Bus
3. Resist = Resistance
4. React = Reactance
5. L_C = Line Charging
6. T_R = Tap Ratio
7. P_SA = Phase Shifter Angle

Actually some identical fields in PSS/E and Tool Box have been differently named. For example, parameters 'resistance' and 'reactance' for *Line* elements in toolbox format are equal to 'R' and 'X' for *Branch* elements in PSS/E format. For an easy understanding, the integrated data structure is defined by adding all the Tool Box data fields to the PSS/E data fields without merging identical fields. Therefore the integrated data set for Busbar is:

Bname, IDE, PL, QL, GL, BL, IA, VM, VA, NAME, BASKV, ZONE, V, A, P_G, Q_G, P_L, Q_L, G_S, B_S, B_T

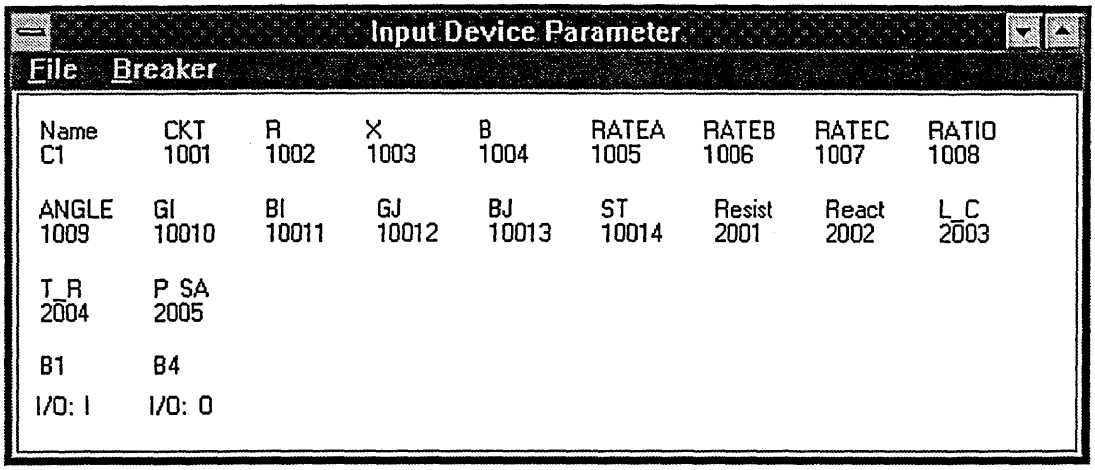
where the first 11 fields is from PSS/E and the rest is from Tool Box. The data sets for other elements are based on the same principle.

2) Create an integrated Power System Project, named as c:\Integrat\Main and then activate this Project

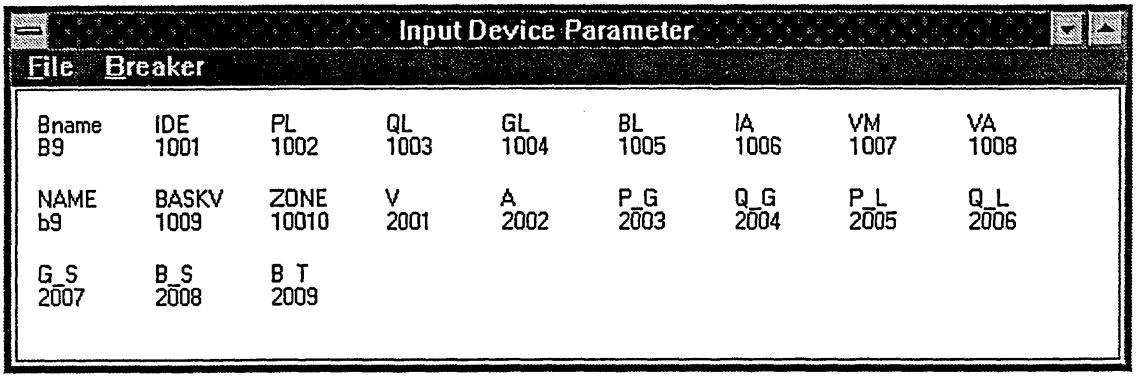


By querying diagram elements the corresponding parameter table can be opened for input parameters. The following are the parameter tables for Cable and Busbar after typing corresponding parameters, where 100X refer to PSS/E data and 200X refer to Tool Box data. The structure of these tables have been given in a format defined by step (3).

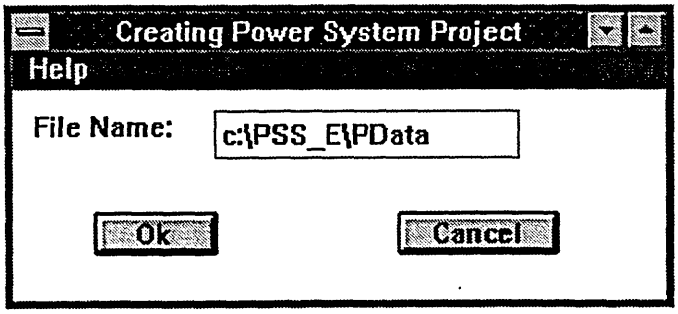
Cable

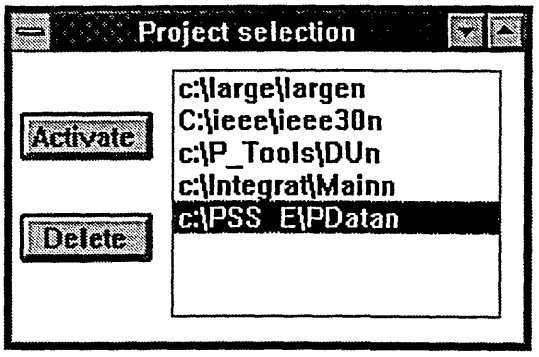


Busbar

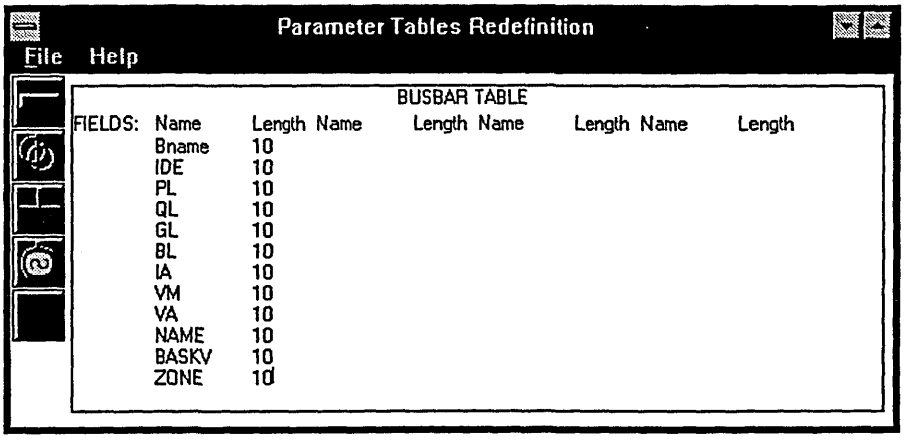


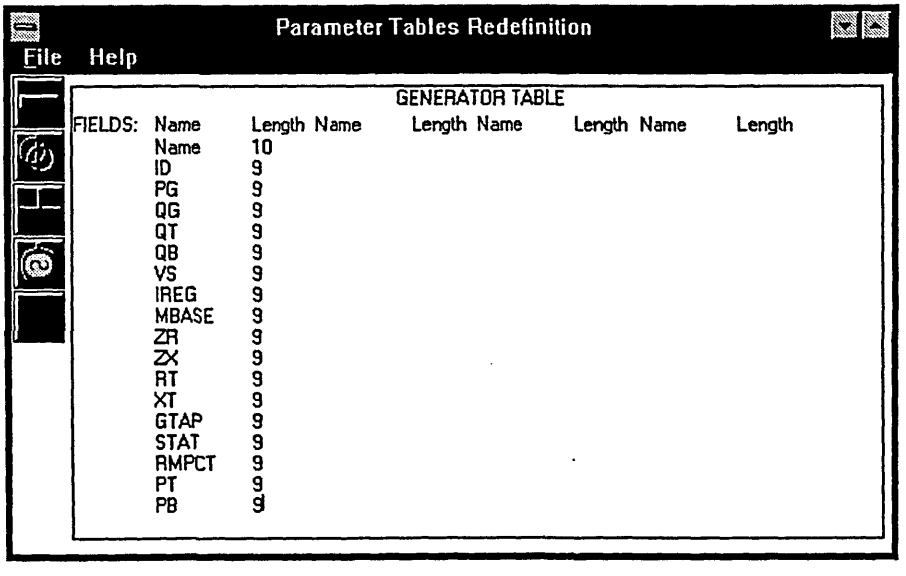
5) Create a Power System Project for PSS/E application, named as c:\PSS_E\PData and then activate this Project.

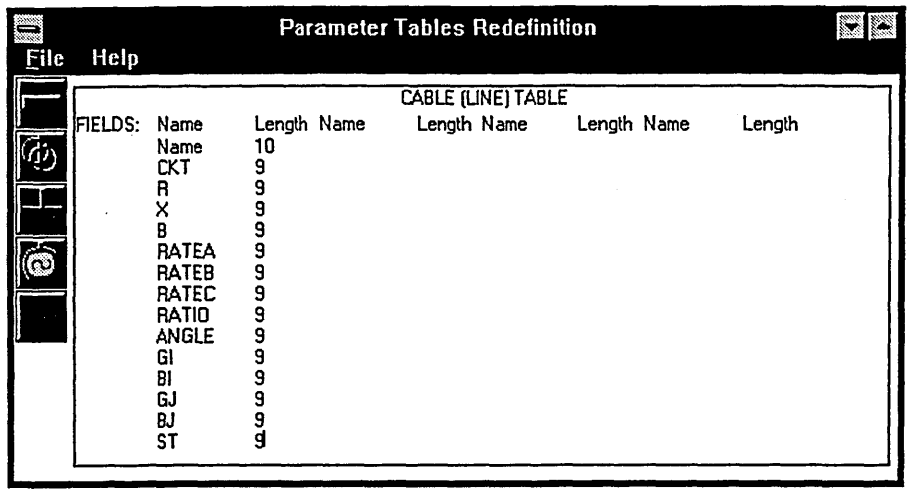


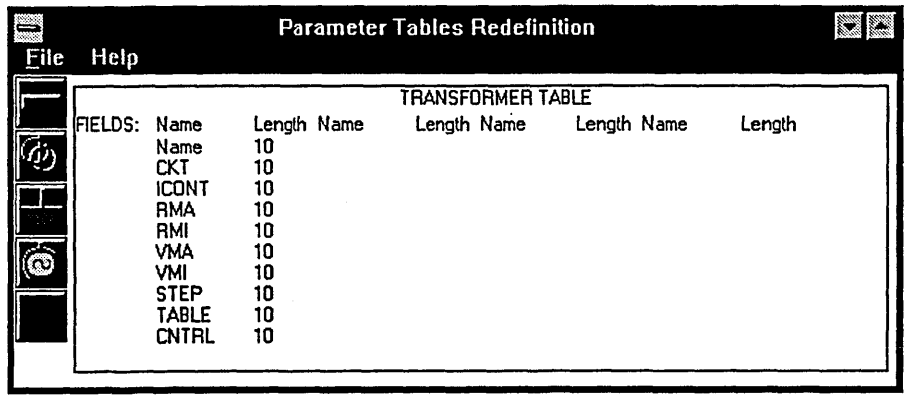


6) Redefine data structure for power system element Parameter Tables according to the data structure required for PSS/E application

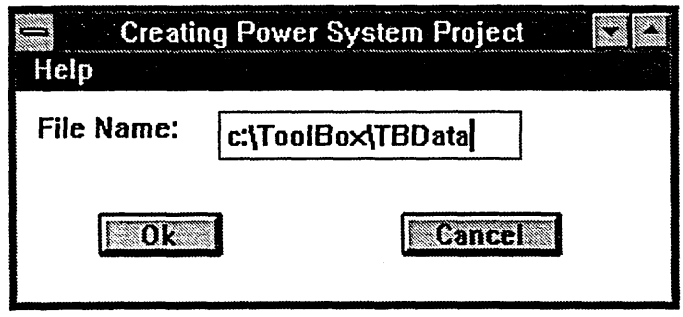


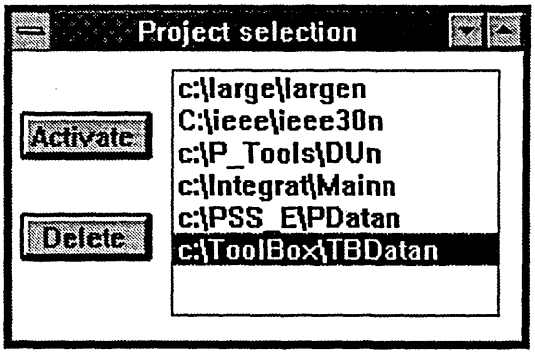




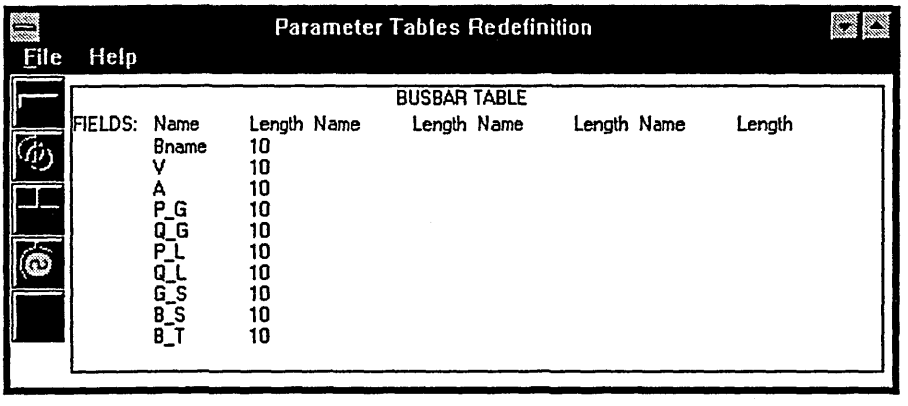


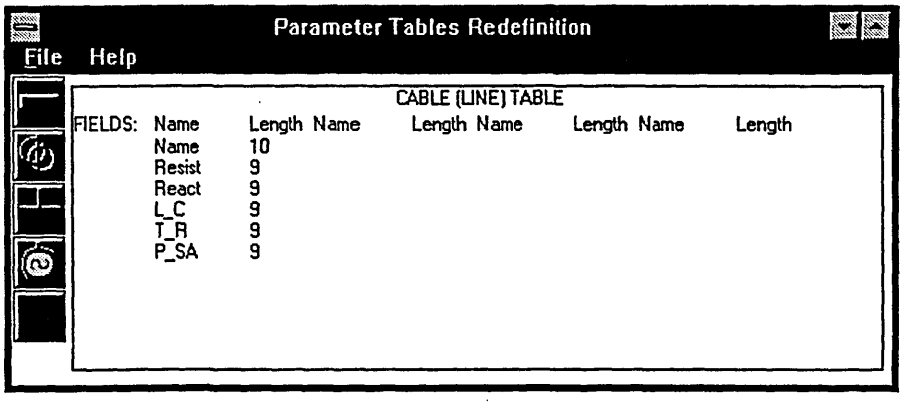
7) Create a Power System Project for Tool Box application, named as c:\ToolBox\TBData and then activate this project



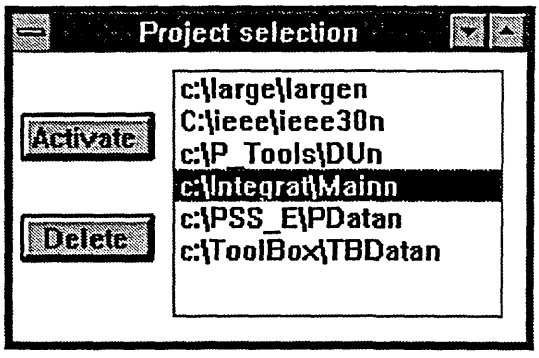


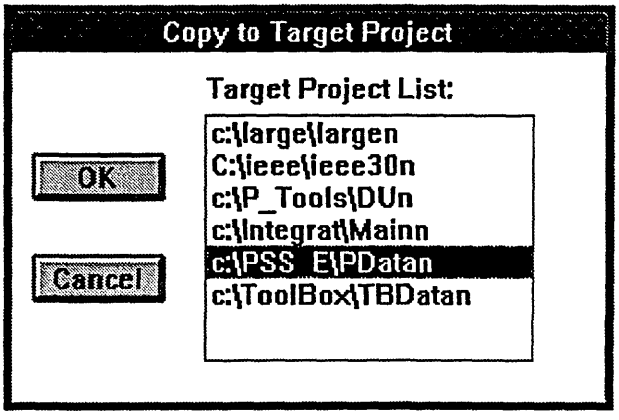
8) Redefine the data structure for power system element Parameter Tables according to the data structure required for Tool Box application

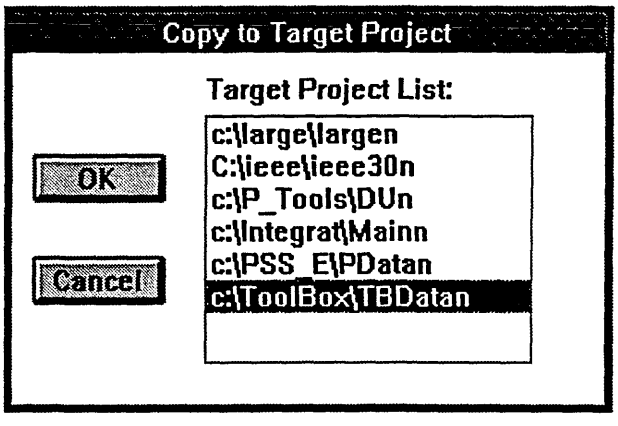




9) Copy the one-line diagram with relevant parameters of the integrated Power System Project to the PSPPSS and PSPTB respectively to obtain individual data sets for PSS/E and Tool Box.

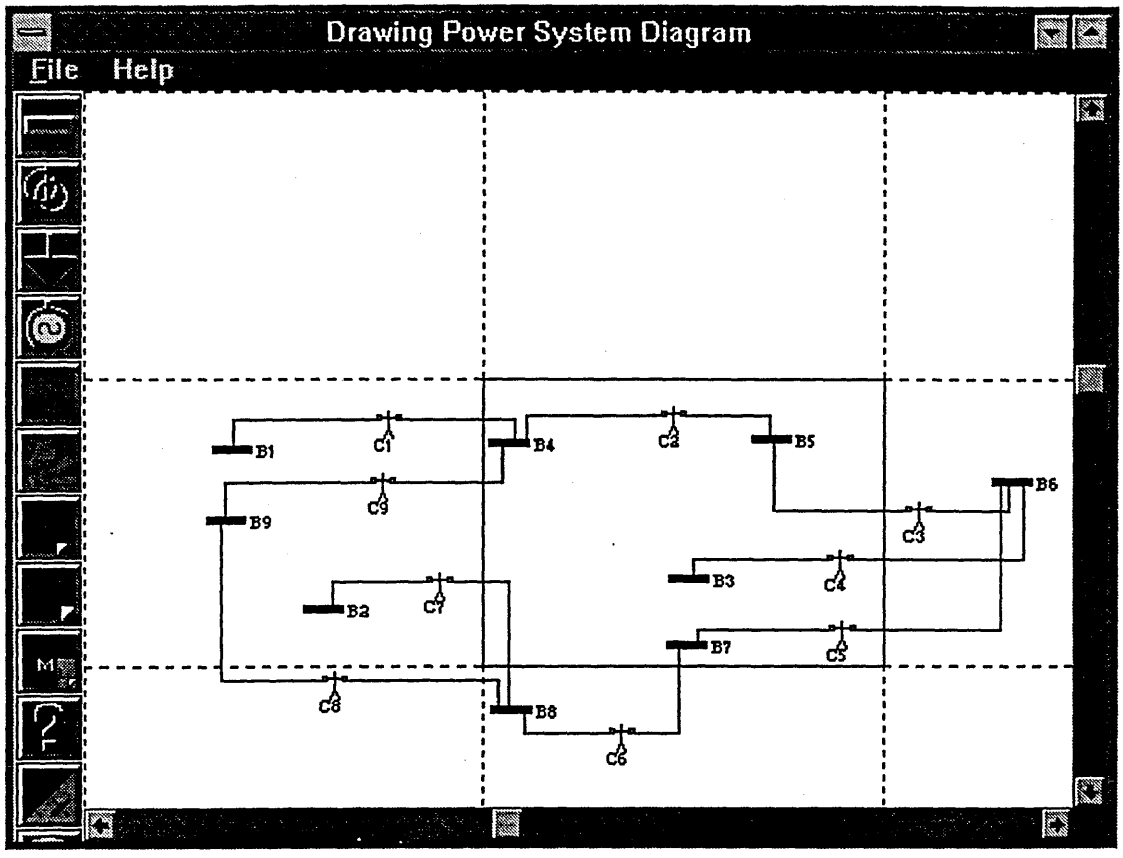




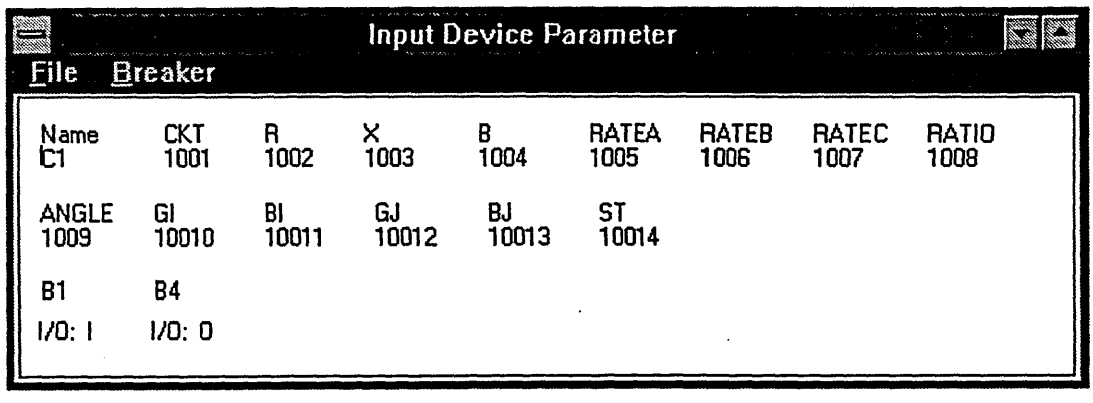


After the above 9 steps, the Project PSS_E\PData and ToolBox\TBData have been completed including one-line diagrams (identical with that of Project Integrat\Main) and parameter tables. The actual result can be found by opening these Project diagrams and parameter tables. This is shown as follows:

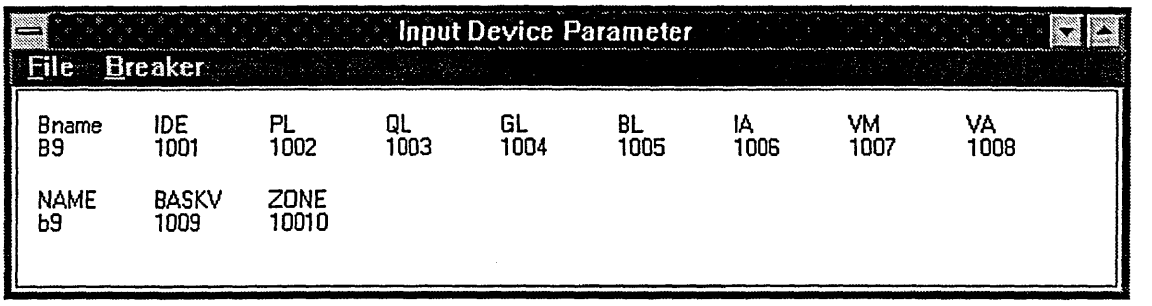
- one-line diagram of PSS_E\PData Project



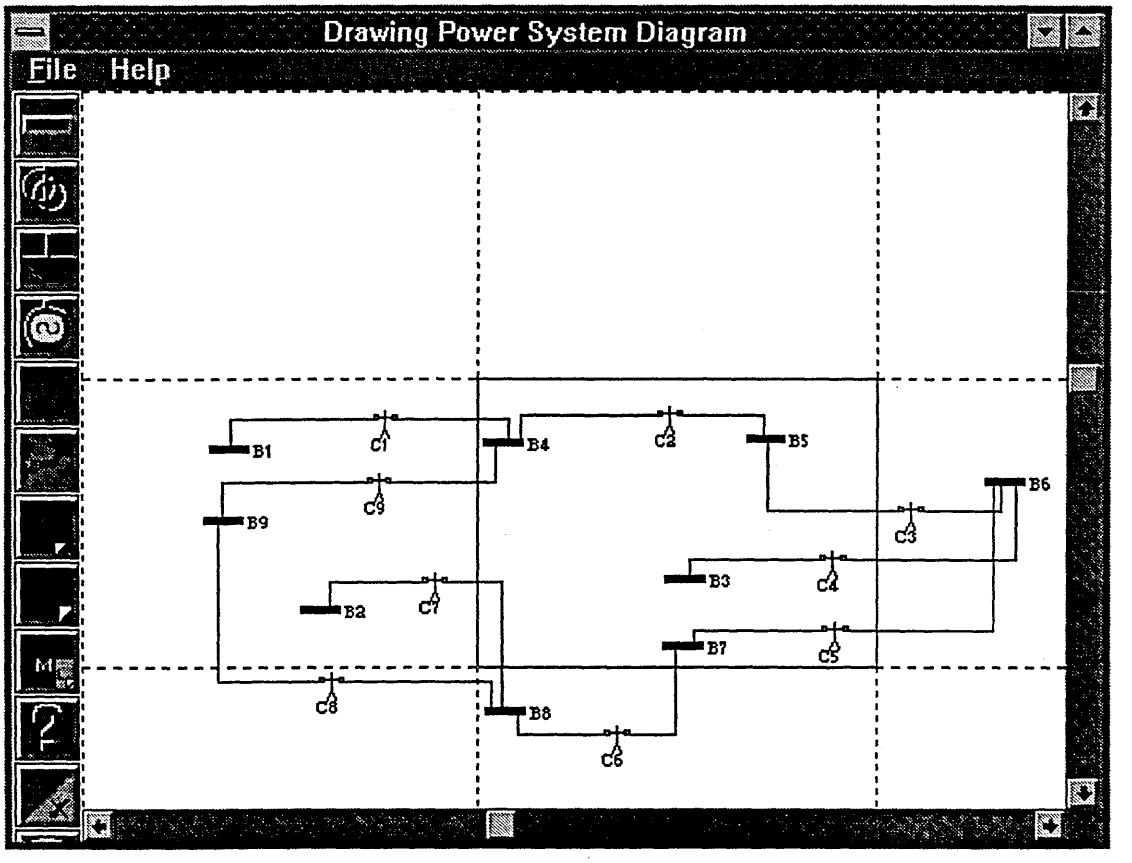
- Cable table of Project PSS_E\PData



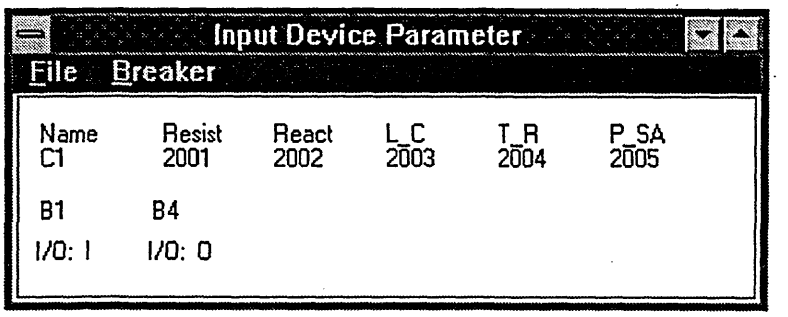
- Busbar table of Project PSS_E\PData



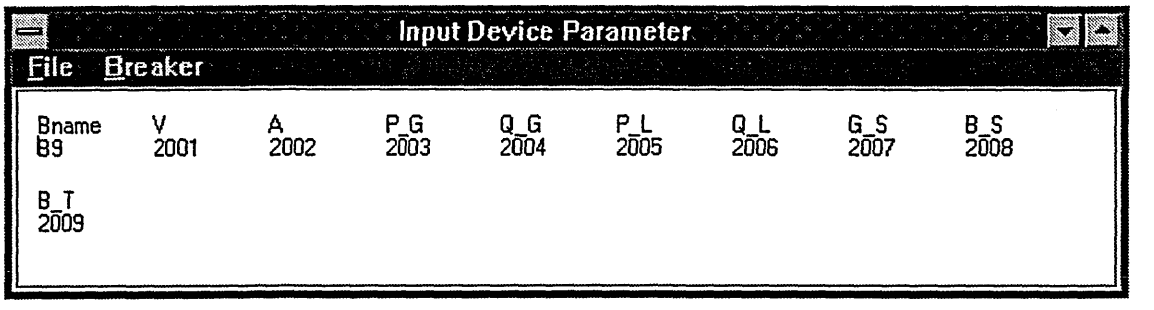
- one-line diagram of ToolBox\TBData



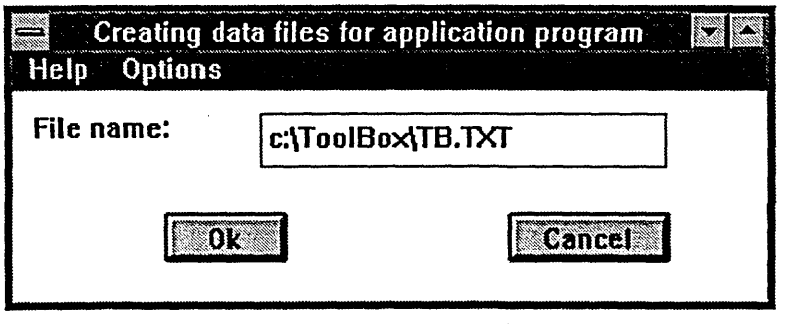
- Cable table of Project ToolBox\TBData



- Busbar table of Project ToolBox\TBData



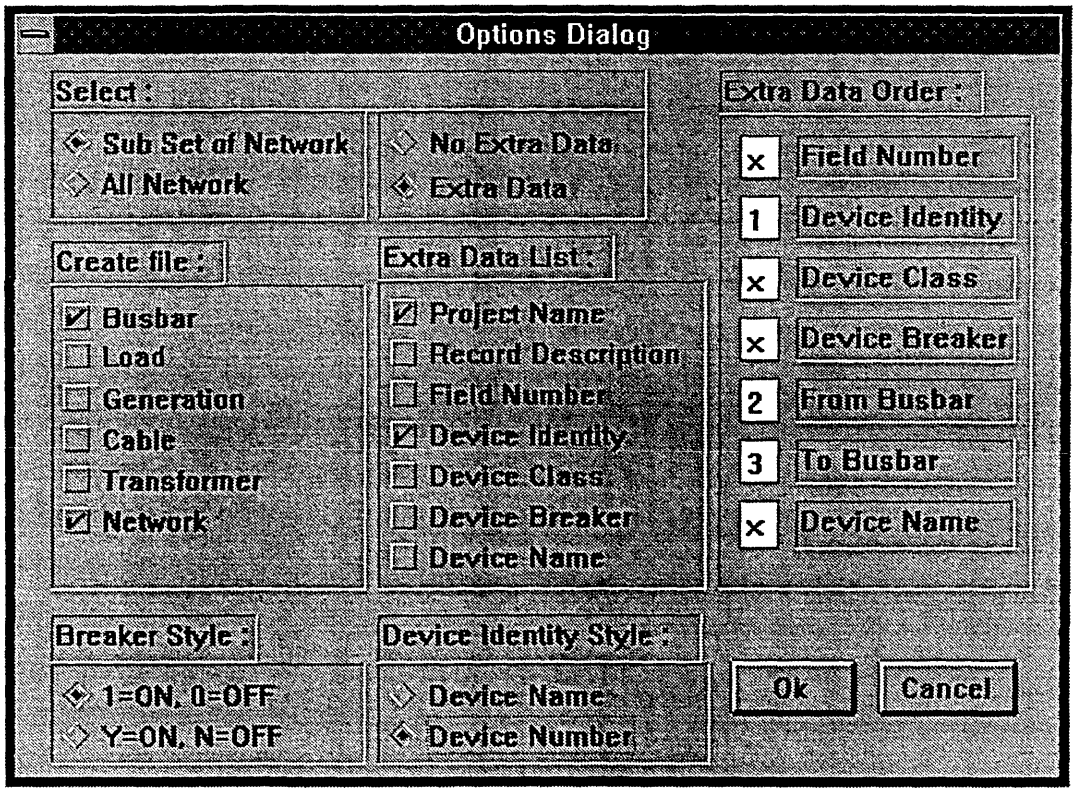
10) Produce ASCII files as the input data files for PSS/E and Tool Box applications. The following is the procedure required for Tool Box, and is similar that required for PSS/E.



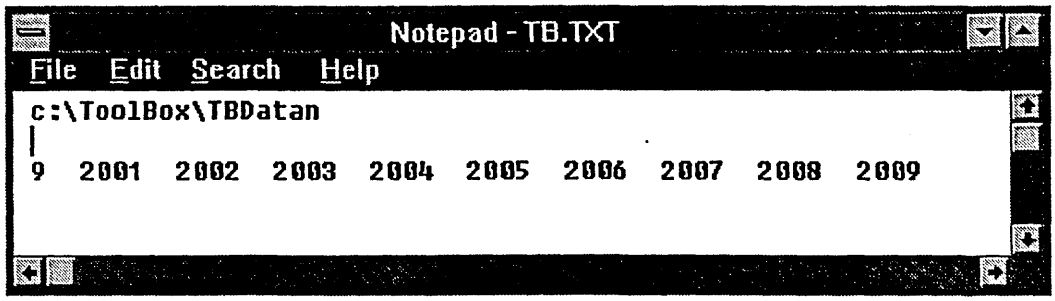
The input ASCII files can be produced according to the user's specification. The following are the procedures for producing Busbar and Cable ASCII files. In a similar fashion an integrated or any other ASCII files can also be produced.

- produce Busbar input ASCII file:

1) Specify requirements and then start the ASCII file producing process

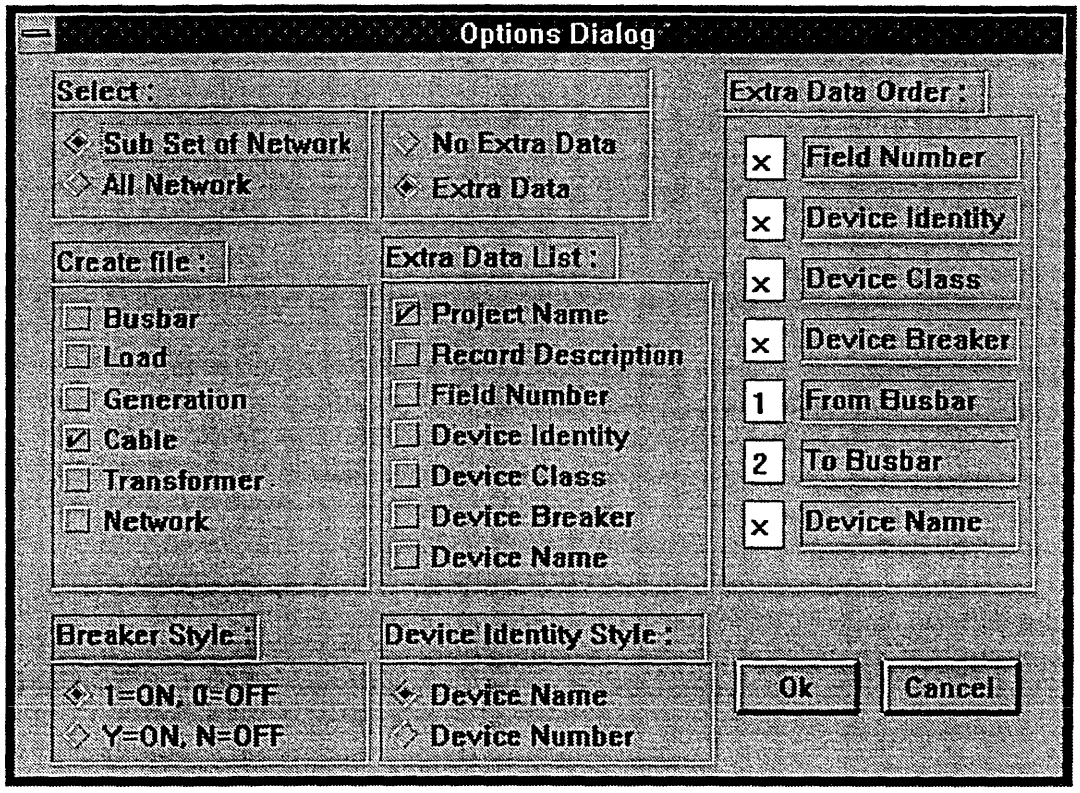


2) The ASCII file produced by Power Shell is located in c:\ToolBox

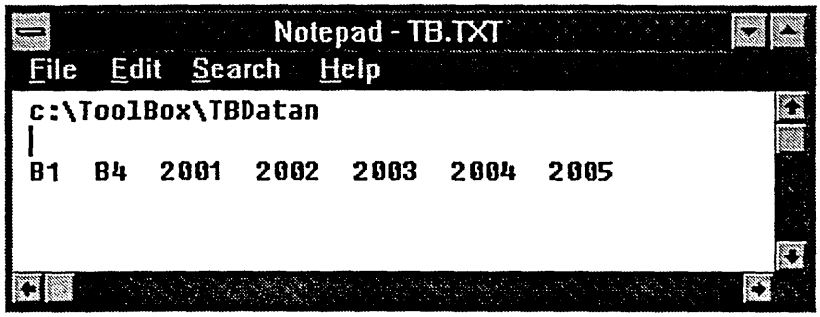


- produce Cable input ASCII file:

1) Specify requirements and then start the ASCII file producing process



2) The ASCII file produced by Power Shell is located in c:\ToolBox



The results have shown that these ASCII files have been produced in the required Tool Box data format, and therefore can be used as the input for the Tool Box calculations.

Using the above test as an illustrative example, the capability of Power Shell for accommodating different power system applications has been demonstrated. In the same way Power Shell can be used for any other power system applications. This capability has, so far, not been realised by other research in this area, even in the products of such leading companies as ICL, PSS/E and ABB. Therefore Power Shell is the first LIME incorporating an open system concept.

7.6 Summary

In this Chapter the Power Shell system evaluation has been made using test data in comparison with the design objectives. These are:

- to enable local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and element parameters for very large scale power systems;
- to provide the user with database data structure redefinition facilities, thereby allowing the accommodation of various power system applications that require different input ASCII file data formats;
- to be compatible with commercial word processing and database systems to benefit users with access to different software tools.

These design objectives have been confirmed at the recent IEE Power Division specialist discussions in November 1994. The Power Shell test results have demonstrated the achievement of the above design objectives which is, effectively, the research objective. Because of the above features such a common information management environment could be significant for bridging the gap between commercial software tools and power system application programs, standardising application program operating environment, and assisting individual off-line work involving professional power system networks.

What should also be stressed is that as a common information management environment, the success of Power Shell has been concerned with only some key issues. Further refinement and improvement of such an information management environment will require considerable time and effort. For example, in ICL, the development of a data management environment for an ICL simulation program has involved 20 persons for 10 years. However such effort is beyond the scope of this research work.

CHAPTER 8

CONCLUSION

To conclude, in this final chapter, the key points of this research and its achievements are summarised. The possibilities for future research related to improvement and extensions of the Power Shell system are also suggested.

8.1 Achievements

This research work has concentrated on the investigation of a Distributed Information Management System (DIMS) for power system applications. The outcome of the research is that Power Shell -a prototype Local Information Management Environment (LIME) of the DIMS has been developed.

Based on a literature survey of the relevant approaches to Information Management Environment for power systems, the LIME development work has been focused on meeting the following requirements:

- to enable local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and parameters for large scale power systems;
- to effect efficient data communications in DIMS;
- to accommodate various application programs requiring different data formats;
- to provide a user-friendly operation environment which is compatible with popular commercial software tools.

The development work has, therefore, involved the analysis and design of the Power Shell data structures, the Power Shell database system, and a method for drawing power system diagrams. Finally, the Power Shell system, as the prototype LIME of the DIMS, has been implemented in MS-Windows, C and Paradox Engine and consists of about 30,000 lines of code (LOC). A number of modern programming techniques

have been investigated and used in the system implementation. These include techniques for multi-job concurrency, Windows applications, drawing diagrams, data flow scheduling, data dictionary, object oriented operation, top-down programming design and modular programming. As a running system, Power Shell has been tested against data from several industrial power system networks including NGC transmission and IEE 30 Bus systems. The test results have been satisfactory for the achievement of the proposed objective.

In summary there are three major contributions of this research. Firstly, a data flow scheduling technique has been developed and used in the Power Shell implementation for removing the limitation of PC internal memory on handling large scale power systems. Therefore in a distributed system, the use of Power Shells as local information management environments enables those local PC platforms to match the host computer on the capability of storing and managing diagrams, network topologies and parameters for large scale power systems. Usually the problems of data over flow have occurred because of the lack of capacity of the PC memory. Thus Power Shell makes possible for the analysis and study of very large scale power systems on a PC platform, which presents an opportunity for individual engineers to have an information management tool to assist their work involving power system networks

Secondly, a method of transmitting only dynamic data has been proposed for efficient communication in the distributed system. Supported by the static data, i.e. diagrams, network topologies and parameters, of local PC platforms, only the data of the current plant status, i.e. dynamic data, needs to be transmitted when a user asks to share the information from the host computer or other local PCs. Thus it has reduced the data

redundancy in the real-time communication required for a DIMS, and therefore achieved an effective way for DIMS communication.

Thirdly, the novel use of database data dictionary techniques has made Power Shell capable of accommodating various power system application programs with different data formats. This means that Power Shell can be used as a standard user interface and data management environment for the development of any power system application programs thus saving developers' effort in the building of operating environments.

In addition the Power Shell system, as an Information Management Environment, provides also quite a few benefits to users for constructing one-line diagrams, storing network parameters and displaying calculation results on the corresponding diagrams. This includes a novel three-layer drawing board with Zoom In/Out facilities for drawing power system network diagrams and geographical maps; automated power system topology data generation based on the diagram drawing; data management support for power system application programs, such as simulation, planning and analysis programs, to be applied for the study of very large scale power system networks. Finally, Power Shell is a PC-based MS-Windows application package and is compatible with those popular word processing tools and database systems. The screen diagrams of power systems produced by Power Shell can be copied to existing commercial software tools such as Microsoft Word and Microsoft Write to generate reports. The data in the Power Shell database tables can be transferred into standard ASCII files that can be further processed by users' application programs. Users can also directly open the Power Shell database tables in a relational database environment such as PARADOX and dBASE IV for data ordering, indexing and analysis. The data output of user application programs or on-line data acquisition

programs can be read back to the database of Power Shell, and then can be displayed on the screen diagram.

All these have demonstrated Power Shell as an open, integrated and universal information management environment for various, including very large scale, power system applications. It has provided a standard user operation and data management environment that is independent of individual application programs but able to serve most of them. This saves the efforts of implementing operation environments for each application program development, and enables the combined use of different data/knowledge processing tools or calculation programs for individual power system applications.

The Power Shell system has proved to be of interest to academics researching in the relevant area. It has been evaluated by, or used for some academic programs in, a number of universities including University of Abertay Dundee (UK), University of Strathclyde, Glasgow (UK), Glasgow University (UK), Domaine University (France), and Nanyang Technological University (Singapore).

In this research work, a number of related publications have also been produced. These include five conference papers [MALI92] [MALI93c] [MALI94a] [MALI94b] [MALI95] and two internal research reports[MALI93a][MALI93b].

8.2 Further work

The Power Shell has provided users with major functions for a standard and universal information management environment for power system applications. Further

improvement, considering the requirements fed back from the Power Shell users, may concern the following issues:

- allow users to define the ASCII data format for Power Shell to read back.

This requirement is due to the large amount of data involved in power system networks. In general, electric power companies have to deal with a large amount of original data contained in the ASCII files of power system historical data. It would be more convenient for the user if the data could be automatically read into the Power Shell database tables instead of being input manually. The difficulty here is, again, that the ASCII file data formats adopted by various application programs are different.

At the moment users are allowed to redefine database table structures for sharing data provided by the Power Shell system. The Data Read Back service provided by Power Shell is based on a pre-designated structure that cannot be redefined by users. This means currently Power Shell cannot read ASCII files with user-defined formats back into the database table. To do so a Data Format Map will need to be built in the Data Read Back program to allow users to specify the ASCII file formats. Then according to the Data Format Map, the Data Read Back program can read and input data to the Power Shell database data tables.

- provide facilities for displaying calculation result charts

In most cases, calculation results for power system application programs have been placed in ASCII files. Displaying ASCII file data on a chart has been required to help visualising calculation results. This may involve some diagram drawing techniques and also communication with users for the ASCII file structure (e.g. via the data format map) and the chart proportion.

- register popular data formats currently used by the Electric Power Industry to Power Shell

To accommodate different data formats, the Redefinition Parameter facilities have been provided by Power Shell. However for users who are not familiar with programming or database systems it might still be a difficult task. Therefore, in addition to Redefinition, the registration of some popular data formats in Power Shell for the user to select could be a more convenient and practical way. This will allow users to select these registered data formats by clicking buttons or items from menus, i.e. just in the way of selecting Font in Microsoft Word. If any other data formats, which are not registered, are specifically required, users can then define the format via the Redefinition facilities. In this revision, difficulties may arise in obtaining the data formats that should be registered since they may be considered confidential to some extent for some companies.

- provide Power Shell users with effective LAN operation utility

A complete LAN function set is now available in the API (Application Program Interface) library of MS-Windows 3.1 for Work Groups and subsequently Window 95. This can be used in the further development of Power Shell, with the support of the Power Shell Project Manager, to achieve automated data categorisation and reorganisation when the local user asks access to the data stored in other platforms, and therefore to provide users with an object oriented operation utility for data communication.

The above further development work can be achieved by using similar programming techniques applied in the Power Shell implementation. In addition further tests for system efficiency on managing information for very large scale power systems may be arranged for optimising data flow scheduling techniques to improve system

performance. The continuous improvement of the Power System and the extended application of the Information Management Environment concept could result in a significant change in the power system application area. Engineers and researchers will be able to work with different power system application programs through standardised operation in a comparatively standard and flexible information management environment. The development of power system application programs will no longer require the implementation of a corresponding operation environment, and will, therefore, be much easier and more normalised.

REFERENCES

[ABB93] ABB, "Power System Computation and Data Management, Functional description", 1993

[ALVAl88] F.L. Alvarado, R.H. Lasseter and Y. Liu, "An integrated engineering simulation environment", IEEE Trans. on Power Systems, Vol. 3, No. 1, February 1988, pp 245-253

[ALDE86] R.T.H. Alden and B. Szabados, "Interactive Fortran for power engineering education", IEEE Transactions on Power Systems, May 1986, pp 160-166

[ANDL92] P.K. Andleigh and M.R. Gretzinger, "Distributed Object-Oriented Data-Systems Design", New Jersey: Prentice-Hall, 1992

[BARL94] M. Barlow and P. Z. Bernat, "The practicalities of implementing on-line network analysis and data management an open system approach," IEE Power Division Colloquium on "Developments with interfacing of power system analysis software with SCADA and data management systems", 15 November 1994, pp 7/1-7/6

[BERR82] Carol Berry, Peter Hirsch and William G. Tuel, Jr., "Data base model for distribution facilities", IEEE Trans. on Power Systems, Vol. PAS-101, No. 2, February 1982, pp 363-370

[BRIT91] Jay Britton, "Utility control centres open to change", IEEE Computer Applications in Power, October 1991, pp 35-39

[BRIT92] Jay Britton, "An open, object-based model as the basis of an architecture for distribution control centres", IEEE Trans. on Power Systems, Vol. 7, No. 4, November 1992, pp 1501-1506

[CANA79] R. Canales-Ruiz, D. Toral Garibay and A. Alonso-concheiro, "Optimal automatic drawing of one-line diagrams", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-98, No. 2, March/April 1979, pp 387-392

[CHAN90] Sherman Chan, "Interactive graphics interface for power system network analysis," IEEE Computer applications in Power, January 1990, pp 34-39

[CHAR92] C.A. Lynch and I.C. Tait, "PSS/E's advanced analytical and graphical techniques in system operation and planning," IEE Power Division Colloquium on "Interactive Graphic Power System Analysis Programs", 20 March 1992, pp 2/1-2/4

[CHER94] Cherry Scientific Software, "Power System Tool Box, User Menu"

[DANE86] M. Daneshdoost and R. Shaat, "A PC based integrated software for power system education", IEEE Transactions on Power Systems, Aug. 1986, pp 1285-1292

[EFTH92] E.N. Dialynas and A.V. Machias, "Interactive power system analysis for design and operations applications," IEE Power Division Colloquium on "Interactive Graphic Power System Analysis Programs", 20 March 1992, pp 1/1-1/6

[GADS92] K. Parton, W. Cartwright and J.B. Gadsden, "The general purpose dinis system," IEE Power Division Colloquium on "Interactive Graphic Power System Analysis Programs", 20 March 1992, pp 6/1-6/5

[GEIS90] K.I. Geisler, S.A. Neumann, T.D. Nielsen, P.K. Bower and B.A. Hughes, "A generalised information management system applied to electrical distribution," IEEE Computer Applications in Power, July 1993, pp 9-13

[GONE84] T. Gonen and J.C. Thompson, "Computerised interactive model approach to electrical distribution system planning," Electrical Power & Energy Systems, Vol. 6, No 1, January 1984, pp 55-61

[GREE92] T.A. Green and A.Bose, "Open systems benefit energy control centres", IEEE Computer Applications in Power, April 1992, pp 45-50

[HAIN94] G. Hainsworth, "Integration of power analysis software into a distribution management system," IEE Power Division Colloquium on "Developments with interfacing of power system analysis software with SCADA and data management systems", 15 November 1994

[HUAN91] J.A. Huang and F.D. Galiana, "A integrated personal computer graphics environment for power system education, analysis and design", IEEE Transaction on Power Systems, pp 1279-1285

[JIAN92] James Jian-Ping Zhang and Roger L. King, "GPSAS- a CAD based tool for power system analysis and simulation," Electric Power Systems Research, 23 1992, pp 31-41

[JOSH90] Vikas Joshi, Kamal Jabbour, Walter Meyer and Frederick J. Ludwick, "Implementation of R&D projects on an Energy management system," IEEE Computer Applications in Power, April 1990, pp 38-43

[KAME92] K. Kamei, Y. Nakamura, S. Abe, S. Takeda, J. Tsukamoto and T. Kaga, "Highly interactive operator workstation for distribution automation

system using spatial data management," IEEE Trans. on Power Systems, Vol. 3, No. 7, February 1992, pp 180-186

[KCHA90] Kevin F. Chan and Jacques Ding, "Interactive network planning and analysis on a personal computer," IEEE Computer applications in Power, January 1990

[KLEI93] Stanley A. Klein and James N. Menendez, "Information security considerations in open systems architectures," IEEE Trans. on Power Systems, Vol. 8, No. 1, February 1993, pp 224-229

[LO86] C.H. Lo, M.D. Anderson and E.F. Richards, "An interactive power system analyzer with graphics for educational use", IEEE Transactions on Power Systems, May 1986, pp 174-181

[MALI92] L. Ma, C.S. Ozveren and L. Crowe, "A distributed problem solving environment for power system planning and analysis," The 27th International Universitys Power Engineering conference, Bath, UK, Sept. 1992

[MALI93a] L Ma, "A Distributed Problem Solving and Decision Support Environment for Power System Planning and Simulation", Research Report, Dundee Institute of Technology, February 1993

[MALI93b] L Ma, "A Distributed Problem Solving Environment for Power System Planning and Simulation", Research Report, Dundee Institute of Technology, June 1993

[MALI93c] L. Ma, C.S. Ozveren, L. Crowe, and K.L. Lo, "Graphically integrated distributed planning and problem solving environment," APSCOM-93,

Second International Conference on Advances in Power System Control, Operation & Management, 7-10 December 1993, Hong Kong

[MALI94a] L. Ma, C.S. Ozveren, L. Crowe, and K.L. Lo, "Heterogeneous LANs as asynchronous distributed planning & problem solving environments," Melecon '94, 7'th Mediterranean Electrotechnical conference, Antalya, Turkey, April 12-14,1994

[MALI94b] L. Ma, C.S. Ozveren, L. Crowe, and K.L. Lo, "Power Shell - An Information Management Environment for Power System Studies", The 29th Universities Power Engineering Conference, Sept. 1994, Galway, Ireland, pp 465-468

[MALI95] L. Ma, C.S. Ozveren, L. Crowe, and K.L. Lo, "A PC-Based Information Management Environment for Large Scale Power Systems", to be presented at IFAC/IFORS/IMACS Symposium LSS'95, July, 1995, London, UK

[MART17] J. Martire and D.J.H. Nuttall, "Open systems and databases", Presented at the IEEE PES Winter Meeting, February 1992

[MOND92] E. Monodon, B. Heilbronn, Y. Harmand, O. Paillet and H. Fargier, "Mars: An aid for network restoration after a local disturbance," IEEE Trans. on Power Systems, Vol. 7, No. 2, May 1986, pp 850-855

[MOON88] H.P. Mooney and J.M. Evans, "A complete relational DBMS for an EMS product," IEEE Trans. on Power Systems, Vol. 3, No. 1, February 1988, pp 325-329

[OCKW92] G.Ockwell and G.Killian, "Configuration management in an open architecture system", Presented at the IEEE PES Winter Meeting, February 1992

[OZVE90] Ozveren, C.S., Bumby, J.R. and Macqueen, C.N. "An Integrated Problem Solving Environment for Distribution System Planning," UPEC '90, pp 485-488

[PAHA91] N. Pahalawaththa, C.P. Arnold and M. Shurety, "A power system CAD package for the workstation and personal computer environment," IEEE Trans. on Power Systems, Vol. 6, No. 1, February 1991, pp 400-406

[PAPA89] M.E. Papadakis, N.D. Hatzjargyriou and D.K. Gazidellis, "Interactive data management system for power system planning studies," IEEE Trans. on Power Systems, February 1989, pp 329-335

[PART94] K.Parton, "Broad principles in integrating network analysis data," IEE Power Division Colloquium on "Developments with interfacing of power system analysis software with SCADA and data management systems", 15 November 1994, pp 2/1-2/4

[PAUL91] G. Paula, "SCADA/EMS: New applications, lower costs, open system", Special Report, Electrical World, July 1991, pp 46-50

[POTT91] Hardy J. Pottinger and Max D. Anderson, "An advanced graphics display capability for power system monitoring and control," IEEE Trans. on Power Systems, 1991

[POWE89] PSS/E, "Program Operation Manual, Section 4"

[POWE94] M. Powell, "The opportunity for real-time power system analysis," IEE Power Division Colloquium on "Developments with interfacing of

power system analysis software with SCADA and data management systems", 15 November 1994

[RAO90] Nutakki D. Rao and S.D. Worthington, "Advanced applications of a relational database manager to power system problems," IEEE Trans. on Power Systems, Vol. 5, No. 1, February 1990, pp 338-345

[RAWL94] M.J. Rawlins "Rationalisation of power system analysis and data management within the National Grid Company," IEE Power Division Colloquium on "Developments with interfacing of power system analysis software with SCADA and data management systems", 15 November 1994, pp 5/1-5/5

[RODO92] A.J. Rodolakis, EurIng R.H. Barnes and Z. Bada "PC based software utilization for power systems analysis and simulation," IEE Power Division Colloquium on "Interactive Graphic Power System Analysis Programs", 20 March 1992, pp 4/1-4/17

[SASS92] A. Mayer Sasson, "Open systems procurement a migration strategy," IEEE Trans. on Power Systems, Vol. 8, No. 2, May 1993, pp 515-521

[SCHL91] Juergen Schlabbach, "AutoCAD Application upgrades power system analysis programs," IEEE Computer Applications in Power, April 1991

[SLI93] S. Li and S.M. Shahidehpour, "An object oriented power system graphics package for personal computer environment," IEEE Trans. on Power Systems, Vol. 8, No. 3, August 1993, pp 1054-1060

[TALU86] Sarosh N. Talukdar, Eleri Cardozo and Ted Perry, "The operator's assistant-an intelligent, expandable program for power system trouble

analysis," IEEE Trans. on Power Systems, Vol. PWRS-1, No. 3, August 1986, pp 182-187

[TREF88] F.J. Trefny, D.P. Gross, B.F. Wollenberg and J.D. O'Hehir, "Integration of an operations information system into an energy management and control system," IEEE Trans. on Power Systems, Vol. 3, No. 1, February 1988, pp 262-266

[VADA93] Subramanian Vadari, Dennis Harding and Larry Douglas, "Open systems for the electric utility industry: What, Why and How?" IEEE Computer Applications in Power, 1993, pp 220-227

[YU89] David C. Yu, Shin-Tzo Chen and Robert F. Bischke, "A PC oriented interactive and graphical simulation package for power system study," IEEE Trans. on Power Systems, February 1989, pp 353-360

APPENDIX

POWER SHELL USER'S GUIDE

Contents

Welcome

Welcome to the Power Shell system, the software that supports large scale power system studies on PC platform with a windows graphical environment. The facilities of drawing power system single line diagrams, the ability to manage power system data and transfer information between the diagram and power system application programs provide a flexible graphical data management interface for your power system simulation, planning and analysis programs.

What's in this manual

This manual consists of the following Parts:

Part I, "**Power Shell Basics**", describes basic Power Shell functions. If you are new to Power Shell, Part I provides an overview to get you started.
Part II, "**Working with Files**" tells you how to start up a Power System Project and work on its diagram and parameters.
Part III, "**Working with Databases**" tells you how to create a new Power System Project and define its data structures.
Part IIII, "**Exchanging Data with Other Programs**", explains how to use data from Power Shell with other power system application programs, and how to use data from other programs with Power Shell.

What you need

The Power Shell runs on any IBM PC 386/486 (or 100 % compatibles) with VGA/super VGA monitor and sufficient RAM and disk storage to build the database. You must also be running DOS 3.0 or later and MS-Windows 3.1 environments.

Installation

Update your CONFIG.SYS file

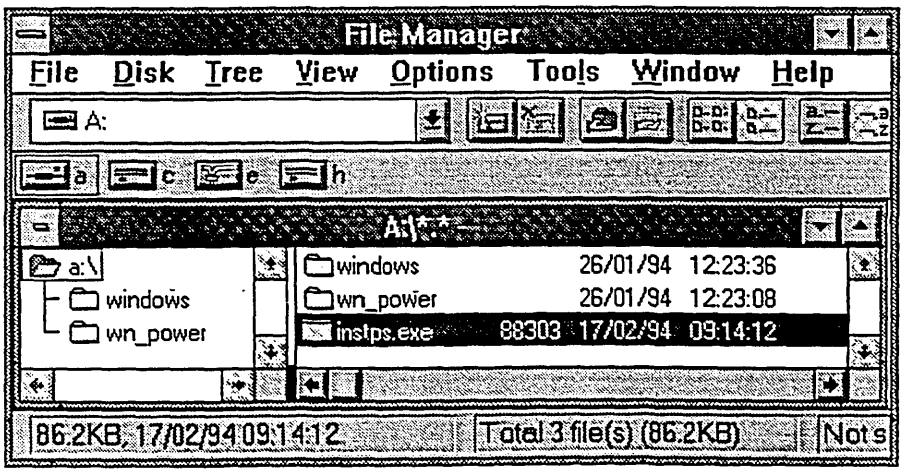
The **CONFIG.SYS** file of your PC system will be automatically updated when you install Power Shell. Alternatively you can update your **CONFIG.SYS** file as follows before the installation.

1. Use a text editor such as MS-DOS Editor to open your **CONFIG.SYS** file. (Your **CONFIG.SYS** file is usually located in the root directory of your hard disk.)
2. Add a **device** command for **EMM386** to your **CONFIG.SYS** file. The **device** command for **EMM386** must come after the command for **HIMEM** and before any commands for device drives that use expanded memory.
   device=c:\dos\emm386.exe 640
3. Disable or remove any other **device** commands for expanded -memory managers.
4. Add an **install** command for **share.exe** to your **CONFIG.SYS** file.
   install=c:\dos\share.exe
5. Save the changes to your **CONFIG.SYS** file.
6. Restart your system by pressing CTRL+ALT+DEL.

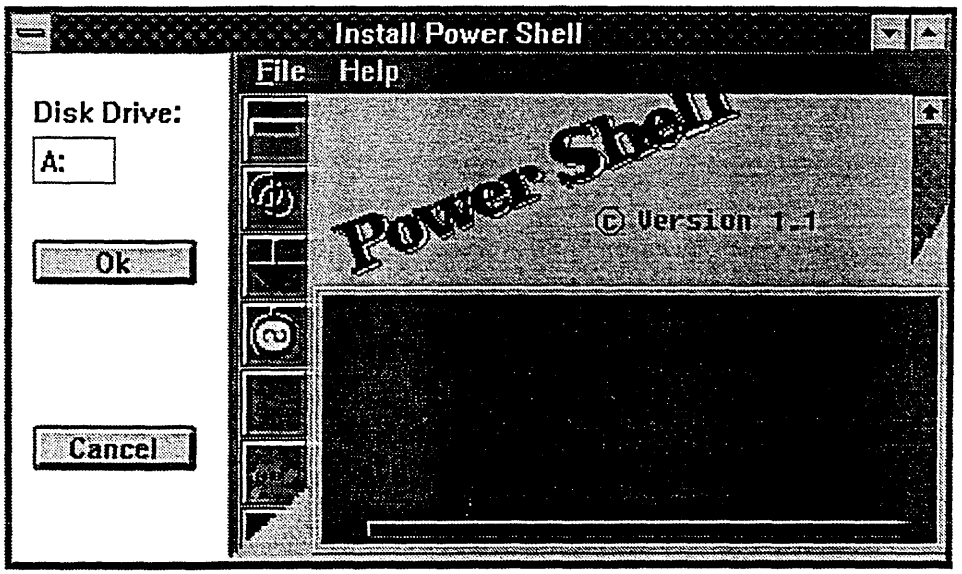
Install the Power Shell on your system

Following these steps to install the Power Shell on your system:

1. Run MS-windows.
2. Insert the Power Shell disk #1 into your source drive **A** (or **B**).
3. Click MS-Window's File Manager and display the files in the **A** (or **B**) drive.



4. Double Click **instps.exe** in the **A** (or **B**) Drive Window to open the Install Window on the screen.
5. Select drive **A** (or **B**).



If you want to install, choose the OK button to start the installation process. Otherwise, choose Cancel to exit the installation process.

To create a Power Shell item by using Window Setup

A Power Shell item will be automatically created after the installation. Alternatively you can follow these steps to create a Power System item for your own application group:

1. In the Main group, choose the Windows Setup icon.
   The Windows Setup dialog box appears.
2. From the Options menu in the dialog box, choose Set Up Applications.
   The Set Up Applications dialog box appears.
3. If you want Windows Setup to search the hard disk for applications, select the Search For Applications option. To specify a particular application, select the Ask You To Specify An Application option.
4. If you selected the Search For Applications option, another dialog box appears. From the Setup Will Search list, select the drive or path you want Setup to search.

5. Choose the Search Now button.
   Windows Setup searches for applications and lists the ones it recognizes in the Applications Found On Hard Disk(s) box.
6. Select the Power Shell item you want to add from the Applications Found On Hard Disk(s) box.
   Click to select the item. (To clear the selection, click the item again.)
7. After you select the Power Shell item, choose the Add button to move it from the box on the left to the box on the right.
   If you change your mind about setting up the Power Shell item after you add it, select it again, and choose the Remove to return it to the box on the left.
8. Choose the OK button.
   For the Power Shell item you selected, Windows Setup adds a Power Shell icon to the Applications group.

How to contact us

The best way to contact us is to:
1. Write a letter with your comments and send it to

Mr Ma Li or Dr C.S.Ozveren
Department of Electronic and Electrical Engineering
Dundee Institute of Technology
Bell Street, Dundee DD1 1HG
UK

2. Send a fax to (0382) 308877 for the attention of Ma Li or C.S.Ozveren.
3. You can also telephone Mr Ma Li or Dr C.S.Ozveren between 9:00 a.m. and 5:00 p.m.(UK time), Monday - Friday.
Tel : (0382) 308000 ext. 2502 (Ma Li), ext. 8260 (Ozveren).

Whichever way you choose, please provide the following information:
- Product name and serial number on your original distribution disk. Have your serial number ready, or we won't be able to process your call.
- Product version number. The version number for Power Shell is in the Power Shell 'About Dialog Box'.
- Computer brand, model, and the brands and model numbers of any additional hardware.
- Operating system and version number.
- Contents of your CONFIG.SYS file.
- The specific steps that reproduce the problem you are having.

Part I Power Shell Basics

Chapter 1 Power Shell overview

Power Shell is a graphical information environment for constructing single line diagrams and storing network parameters of power systems, and is intended to facilitate large scale power systems studies for both academic research and industrial applications. It is currently running under the MS-WINDOWS environment on PC platforms.

Power Shell is composed of a relational database and a graphical operating environment. The main features and functions of Power Shell include:

Large scale power system modelling support

You can use Power Shell to construct single line diagrams of large scale power system networks and record corresponding parameters. For Power Shell the restriction on the scale of power system depends only on the capacity of the PC hard disk rather than the internal memory.

Open architecture for data exchange

Power Shell supports data access to other software programs in four ways:

- The screen diagrams of power systems produced by Power Shell can be copied to existing commercial software tools such as Microsoft Word and Microsoft Write to generate reports.

- You can directly open the Power Shell database tables in a relational database environment such as PARADOX and dBASE IV for data ordering, indexing and analysis.

- Power Shell can transfer the data from the database tables into standard ASCII files that can be further processed by your application programs. These ASCII files can also be transferred to different kinds of computer platforms such as APOLLO and SUN workstations. The ASCII files produced by Power Shell contain the following information on a power system network:

    - network topology (the connection of elements);
    - circuit breaker status;
    - data and parameters of network elements.

- The data output of your application programs or on-line data acquisition programs can be read back to the database of Power Shell, and then can be displayed on the screen diagram.

Flexible data management method

To meet the requirements of various research works a flexible data management method is used by Power Shell. You can redefine the database data structure of Power Shell. For each element of the power system network, you are allowed to define up to 80 data fields. In the Power Shell system operation the database tables and ASCII files are formatted automatically to match the defined data structure.

Convenient operation

Power Shell supports you in the management of power system networks as individual projects through a Project Manager module. With the help of the Project Manager, you will be in an object oriented operating mode. When a Power System Project is specified, all the procedures such as drawing diagram elements, updating parameters and switching circuit breakers, will automatically focus on this project as the currently active object.

Diagram interface facilities

- A three-layer drawing board

A size-unlimited proportional three-layer drawing board is provided for you to draw and display power system network diagrams at levels of High Voltage, Middle Voltage and Low Voltage. The elements of different layers can be connected according to their relationships. The diagrams in these three layers

can also be proportionally overlapped to display a global diagram of the power system network. There are six available symbols for drawing diagrams. These are:

- Busbar,
- Cable(Line),
- Transformer,
- Generator,
- Load,
- Map-line.

Meanwhile Zoom In/Out and Move facilities are provided with the three-layer drawing board for users to switch current operation layers.

- Data inquiry and update

Through the diagram interface you can designate a certain element of the power system network, open the data table of the element and update its parameters.

- Power system network partitioning

Power Shell allows you to partition the power system network logically. The information of the partitioned sub network can be transferred to ASCII files.

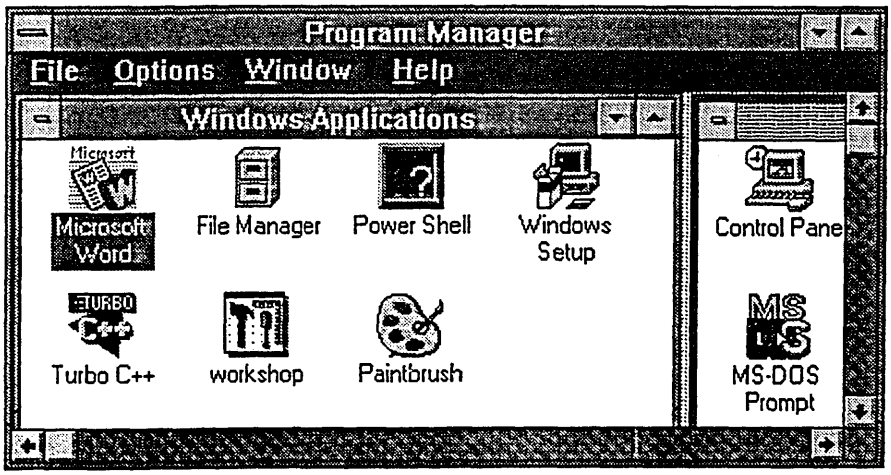
Power Shell provides an opportunity to build visible relationships between the elements data and the diagram of a power system network. With the data support of Power Shell other power system application programs such as simulation, planning and analysis programs can be applied to the studies of large scale power system networks.

Chapter 2 Introduction to Power Shell functions

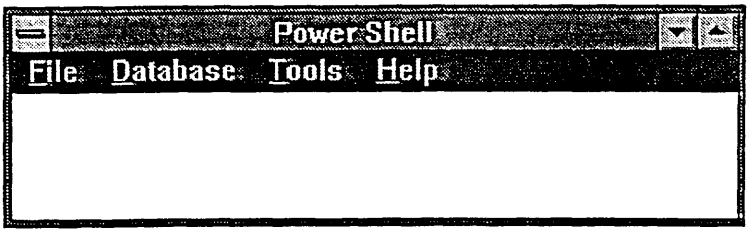
This chapter introduces the basic structure of the Power Shell menus, and explains each item in the menus.

Power Shell menus

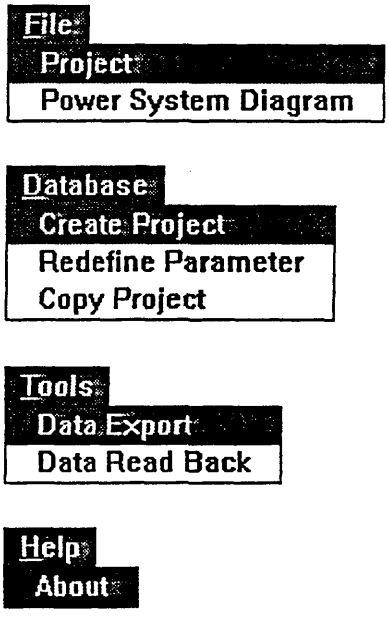
You will find the Power Shell icon in the Program Manager window in MS-Windows environment.



To start Power Shell, click the Power Shell icon. The Power Shell root window appears on your screen.



There are four root window menus.

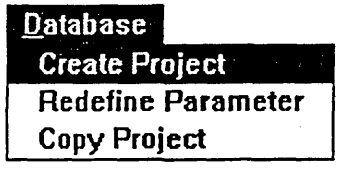


You can open any of these menus to choose an item from the selected menu.

Each item in these menus can provide service for data operation except the option About displaying the information of product support. The following are the detailed explanation of these root widow menus.

Database operations

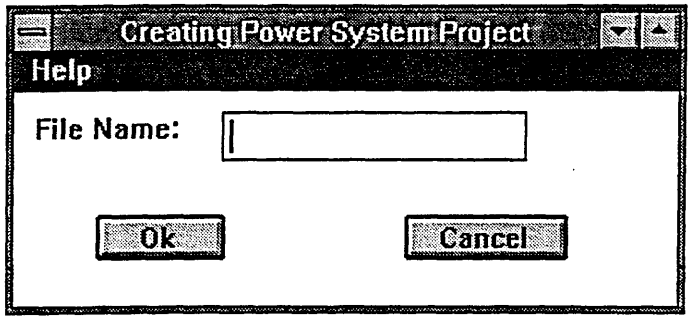
The Database menu contains three items: Create Project, Redefine Parameter and Copy Project.



Projects creating and registration

Power Shell can manage several Power System Projects. Each Power System Project is composed of seven data base tables which are logically related and

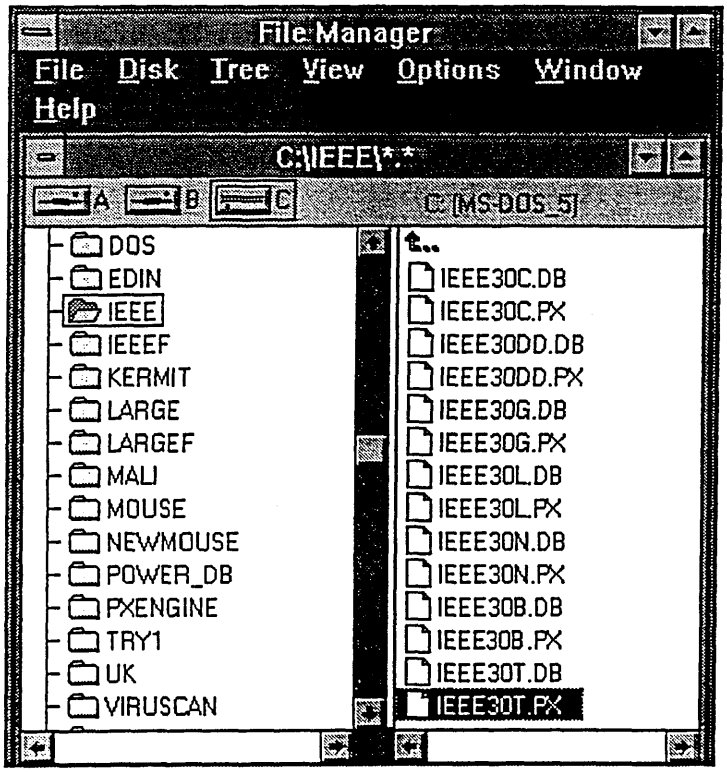
seven index files. A Project Manager is used to register all of Power System Projects. Create Project from the Database menu is a tool for users to create new Power System Projects.



The Power System Projects created will be automatically registered to the Project Manager. The seven database tables for this Power System Project will be produced under the directory you designated in the hard disk. The database tables of the Power System Project are named by Create Project according to the following rules:

ProjectNameN.db	Network configuration table
ProjectNameN.px	Primary index file of the Network configuration table
ProjectNameB.db	Busbar table
ProjectNameB.px	Primary index file of the Busbar table
ProjectNameL.db	Load table
ProjectNameL.px	Primary index file of the Load table
ProjectNameG.db	Generator table
ProjectNameG.px	Primary index file of the Generator table
ProjectNameC.db	Cable(Line) table
ProjectNameC.px	Primary index file of the Cable (Line) table
ProjectNameT.db	Transformer table
ProjectNameT.px	Primary index file of the Transformer table
ProjectNameDD.db	Drawing Data table
ProjectNameDD.px	Primary index file of the Drawing Data table

Network configuration table is for recording relationships of power system elements; Drawing Data table is used to record diagram data of power system elements; Busbar table, Load table, Generator table, Cable (Line) table and Transformer table are the tables of the data records for the power system elements parameter.



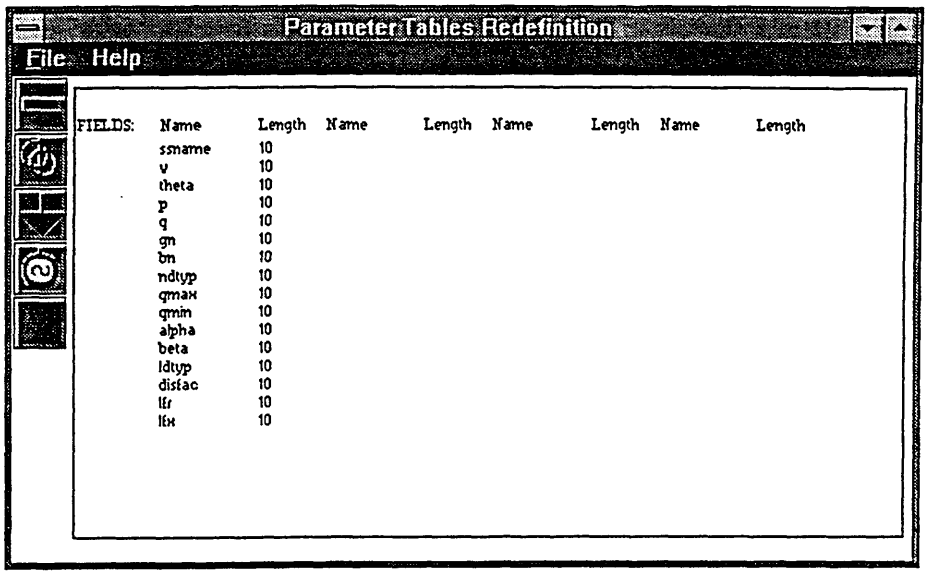
Power Shell can manage a number of Power System Projects created by the users. When you want to work on an existing Power System Project, you must activate the Power System Project to make it a Current Project through the Project option from the File menu in the root window. For more details see the section on Activating and deleting project in this Chapter and also see Chapter 3 Managing project.

Once a Power System Project is built, default data structures of the Busbar table, Load table, Generator table, Cable(Line) table and Transformer table are defined. You can redefine these data structures through the Redefine Parameter option, which is further explained in the next section on Data structure operations, and more details are found in Chapter 6 Redefining parameters.

The database tables built by using Create Project are compatible with PARADOX database table. You can use PARADOX as a tool to manage the data in the database tables of Power Shell when necessary.

Data structure operations

Power Shell allows you to redefine data structures for parameter tables through the Parameter Tables Redefinition window.



When a Power System Project is created through the Create Project option, seven database tables are produced in the hard disk. These seven tables are used to record descriptional data for different aspects of the power system network, which include five element-parameter tables:

Busbar table,
Load table,
Generator table,
Cable(Line) table,
Transformer table.

A default definition of the name and the length of each field in the table is given by the Create Project according to a normal routine. For the above five element-parameter tables, you can redefine the name and length if you require a different structure. In the redefinition operation, you can add or delete the fields in the table and change names and lengths of the fields. After the redefinition procedure new data table based on the structure redefined will be produced by Redefine Parameter. The name of the new data table will not be changed, and still belongs to the original Power System Project. Since the redefinition effects the structure of the data tables, when a data table containing some data is

redefined, the data will be deleted by Power Shell. This means that you always obtain a empty table after redefinition. For each data table you can redefine up to 80 fields. The field lengths are restricted to 24 characters. The first field of the data table is the element name that is the relational keyword fixed by the Power Shell system and cannot be changed by users.

The database table operating interface of Power Shell is always formatted automatically to match the latest data structure. The structure of data tables will also be transferred to ASCII files through the Data Export option in the Tools menu. The details are found in the Data Export operations section in this Chapter and also in Chapter 8 Exporting data.

Project data copy function

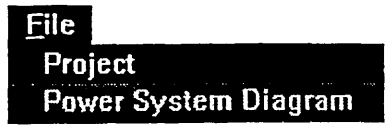
The project data copy function is for help create a new Power System Project without redrawing and rewriting the diagram and the data which are existing. Through the Copy Project option you can create a new Power System Project by copying the diagram and the relative data of the Current Project, and then inputting rest of data for the new Power System Project.

In the project data copy process, the data that will be copied from the Current Project to the target Project, i.e. the new Power System Project (or any other Power System Project to which you want to copy the data from the Current Project), is selected according to the data category and structure.

Current Project Data	Data that will be copied to the target project
Network configuration data Drawing diagram data	all
Busbar data Cable/Line data Transformer data Load data Generator data	if there is the same field name and field length in the target project

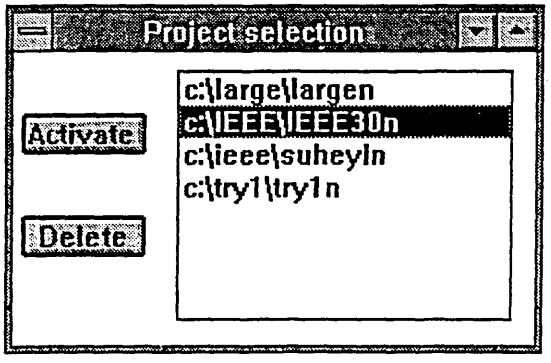
File operations

The File menu contains two options: Project and Power System Diagram.



Activating and deleting projects

When you choose the Project option, a Project Selection window will appear. Through this window you can activate or delete a Power System Project.



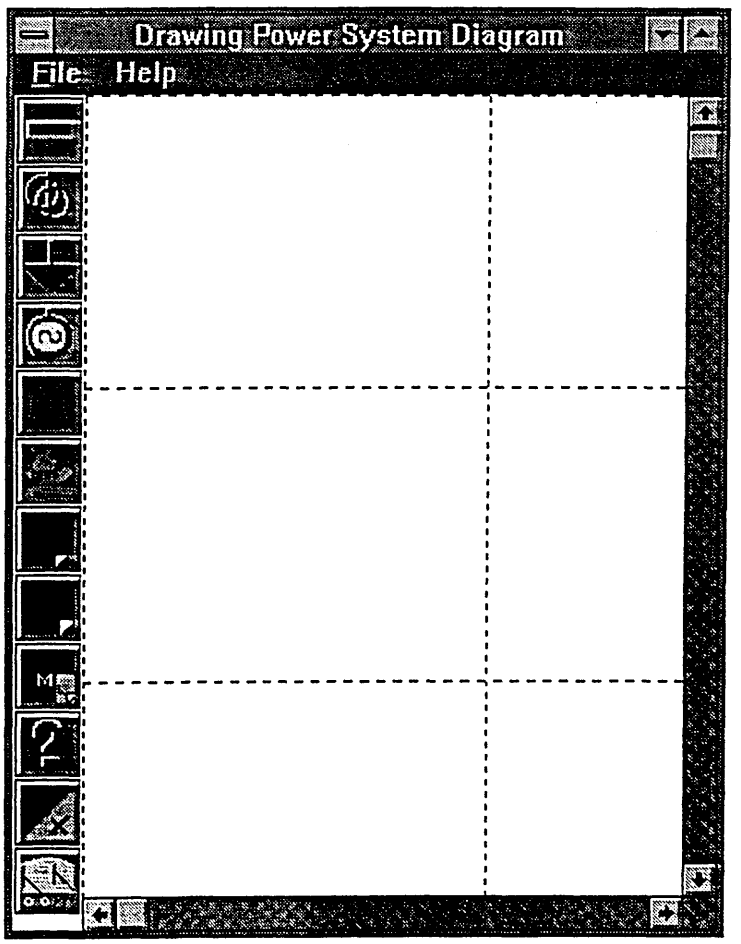
Power System can manage a number of Power System Projects. Each Power System Project, as a set of data and relations, is composed of seven data tables and seven index files. These tables and files are used for the description of the connections of the power system elements, the records of the elements data and the information of the power system single line diagrams.

In Power Shell a Project Manager module is employed to help you in the management of the database table. Project Manager is the operating scheduler. In Power Shell operation, no operation procedures can start until the tasks have been assigned by the Project Manager. When you choose the Project option, the Project Manager will be notified and provides a Project Selection window for you to specify and activate or delete a Power System Project. An activated Power System Project will stay active, as the Current Project, until another Power System Project has been activated. With the support of the Project Manager, you are presented with an object oriented operation style. When you specify a Power System Project, all the operation procedures will focus on this Power System Project as the currently active object. So all you need is to choose operation options rather than to manage data tables.

The procedures scheduled by the Project Manager include: Power System Diagram, Redefine Parameter, Data Export, and Input and Update Parameter. These are detailed in the corresponding sections in this manual.

Diagram and parameter operations

When you choose the Power System Diagram option, an operating environment of Drawing Power System Diagram (DPSD) will appear.



DPSD is a multi-function window that allows you to

- draw/update power system single line diagram,
- input/update the data of power system elements,
- access the data of power system elements,
- zoom in/out power system diagrams,
- update power system breaker status,
- partition the power system network.

DPSD provides six drawing symbols. These are:


Busbar


Load


Generator


Cable(Line)


Transformer

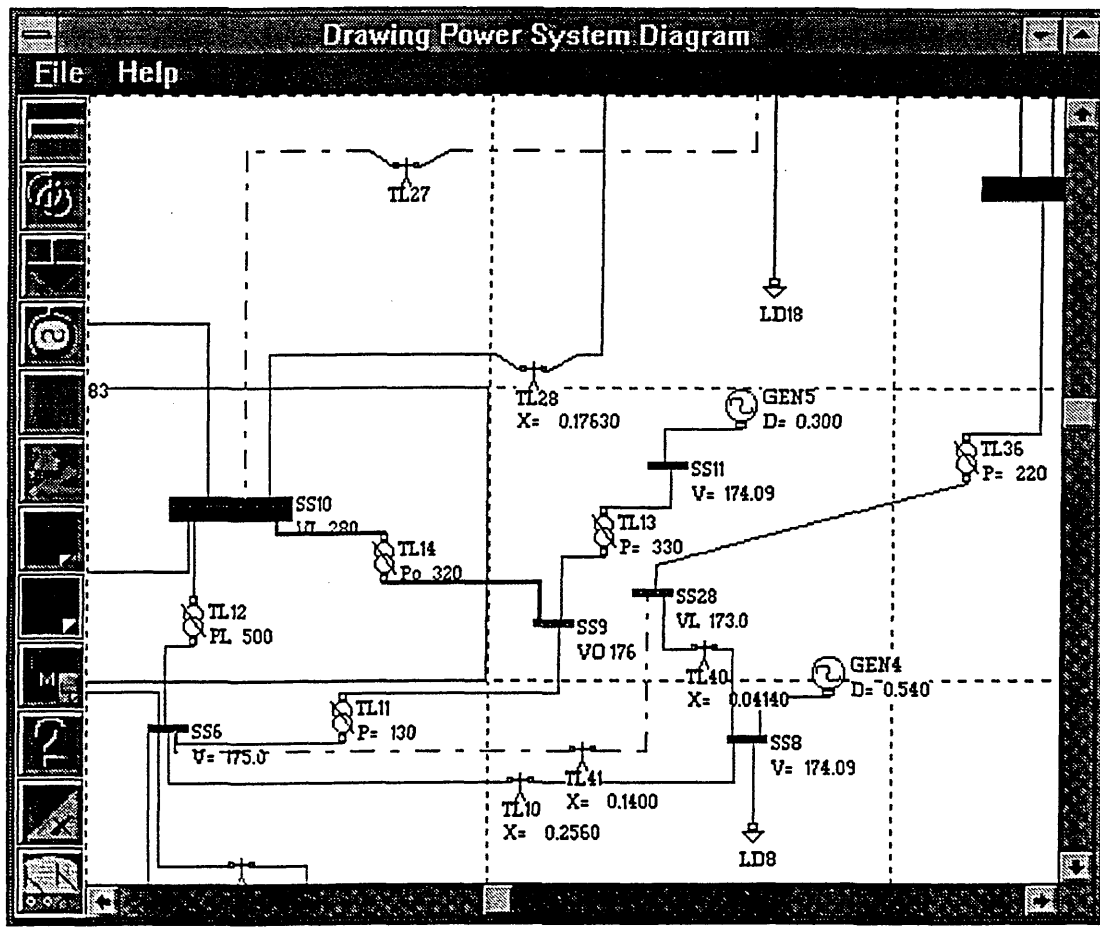

Map-line

Using these six symbols you can draw a single line diagram of the power system network topology and also that of a distribution network with geographical background.

When a single line diagram is being produced, the drawing information is automatically written to the Current Project table by DPSD. The information recorded can be categorised into two groups. One is of the relationships of power system elements, which are recorded in the Network configuration table; the other is of the diagrams and coordinates, which are recorded in the Drawing Diagram table. If any element has been drawn, DPSD will open the input window of the corresponding database table according to the element category. Through this window you can input relevant data.


DPSD window provides you with a drawing board with three layers: High-Layer, Middle-Layer and Low-Layer. The proportion of these three layers is 1:3:9. You can draw the network at High Voltage, Middle Voltage and Low Voltage levels, all of which contain different amount of elements, on High-Layer, Middle-Layer and Low-Layer respectively. You can also connect the elements in the different board layers according to the elements relationships. If you need to relate the network diagram to a geographical distribution, the effective way is to draw the geographical map at the High-Layer of the drawing board and draw the single line diagram of the power system network at the Middle-Layer and the Low-Layer of the drawing board.

You can overlap the diagrams drawn on the two or three different layers in the DPSD window proportionally and move the layers synchronously.

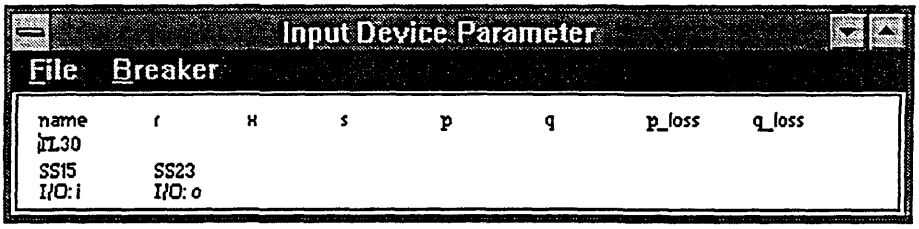


To do so you need to use the Zoom In/Out and Move facilities. For more information see the section of Drawing power system diagrams in Chapter 4.

The size of the drawing board depends upon the hard disk capacity rather than the internal memory. If your PC hard disk has enough capacity, Power Shell will allow you to draw very large scale power system networks.

DPSD has a background service window. When you require to access and update the data of power system elements, this server window will come to

the foreground to provide you with a database table operating environment for you to input and update device parameters.



Input and Update Parameter (IUP) is a service program that cannot be driven directly through the menus in the Power Shell root window. It is a service procedure for the DPSD window. The execution and operation of IUP is controlled by DPSD and the Project Manager. IUP provides you with an operating interface of data tables that are formatted in the same data structure with that of the element data tables in the database. The structure of the IUP data table interface will be updated automatically as you redefine the data structure through Redefine Parameter.

The operating objective of IUP is provided by DPSD and the Project Manager. The Project Manager guides IUP to focus the Current Project and DPSD notifies the category of the elements data tables. Through IUP you can input, update and display the data of the power system network, and change the breaker status of network elements. Information about any operation that you have done with IUP will be written back to the database tables.

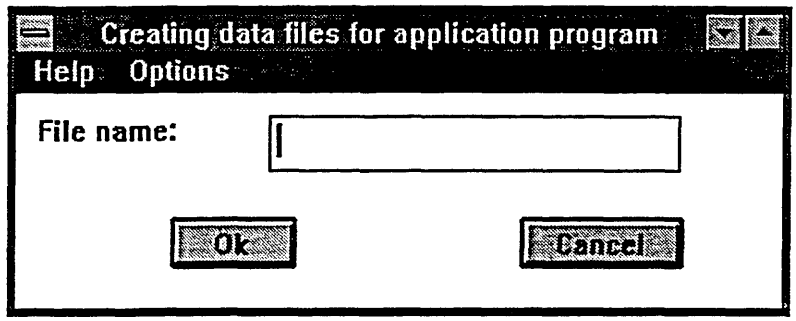
 DPSD also provides you with a Mark function to mark network elements for a logical partition of the network diagram. The data of the marked elements can be exported to ASCII data files, but unmarked elements will be ignored. By the Mark function you can specify any part of the power system network, and obtain information of the specified part.

Tools for data exchange

The Tools menu contains two options: Data Export and Data Read Back.

Data Export operations

Data Export is a data exchange channel between Power Shell and your application programs. Through Data Export you can create data ASCII files and transfer data from the Power Shell database to the ASCII files.



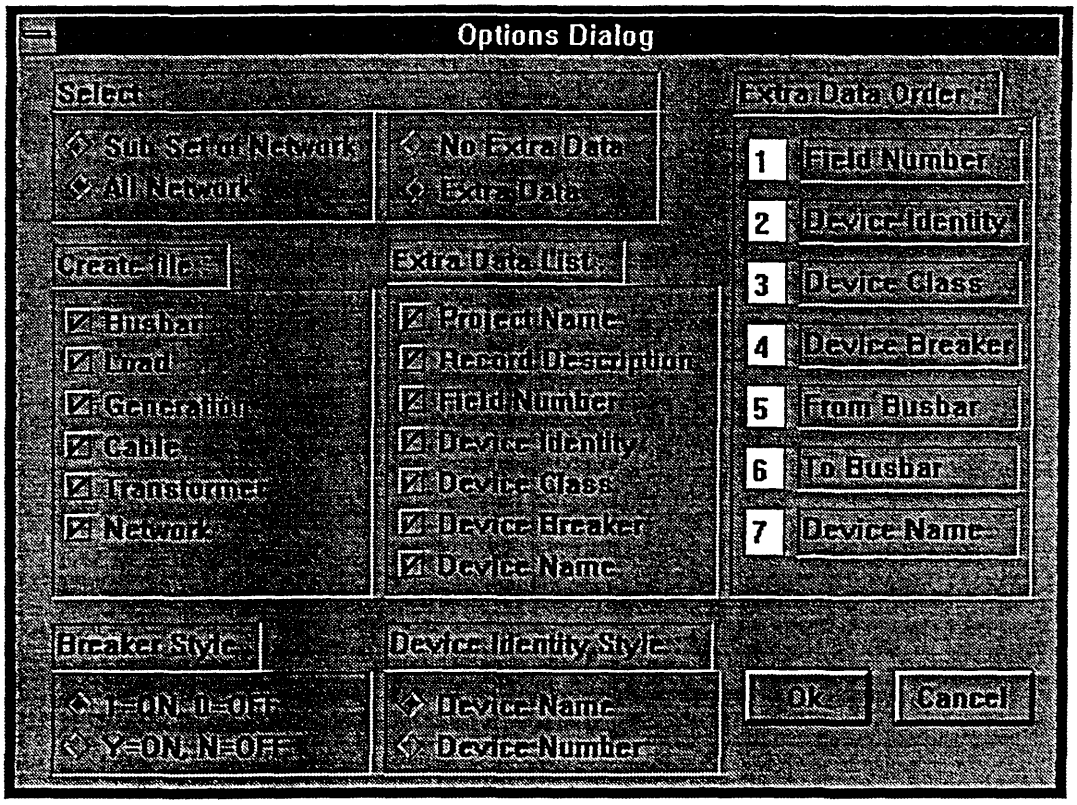
The operating object of Data Export is designated through the Project option from the File menu in the Power Shell root window. The ASCII files output from Data Export may include the information of either

- the whole power system network, or
- a part of power system network.

When the information of a part of the power system network is required to be exported, Data Export will generate ASCII files according to the network elements marked by users in the Drawing Power System Diagram window. The ASCII files generated by the Data Export include:

- Busbar file
- Load file
- Generator file
- Cable (Line) file
- Transformer file
- Network file

You can determine whether and which of the above files need to be generated, and which data, such as network elements data, breaker status and the relationships between elements, should be included. The elements data in the ASCII files has been formatted according to either the default data structure of the database tables in Power Shell if the data structure has not been redefined or, otherwise, the redefined structure.

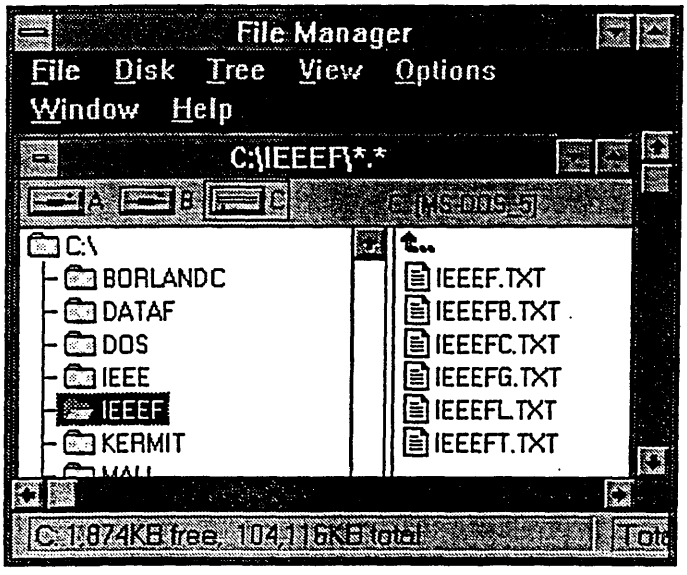


The six files of the Current Project will be produced under the designated directory in the hard disk. These are named by Data Export according to the following rules:

ProjectName.txt	Integrated file;	ProjectNameG.txt	Generator file
ProjectNameB.txt	Busbar file;	ProjectNameC.txt	Cable (Line) file
ProjectNameL.txt	Load file;	ProjectNameT.txt	Transformer file

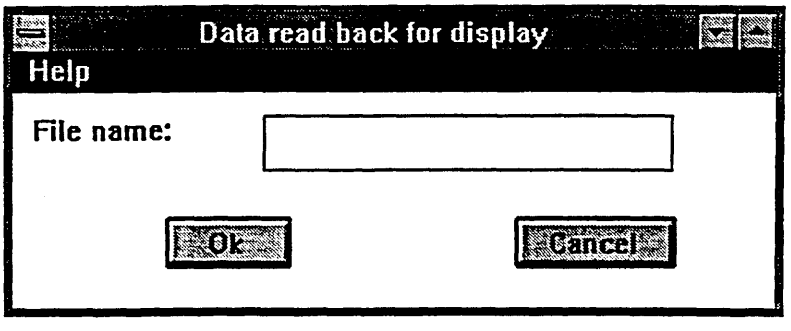
where the Integrated network file is for recording connection relationships and the data of power system elements; Busbar file, Load file, Generator file, Cable (Line) file and Transformer file are the files of the parameter records for the corresponding power system elements.

The element data records in the ASCII files have been formatted according to either the default data structure of the Power Shell database data tables with the default extra data if no redefinition has been made or, otherwise, the redefined data structure and selected extra data in a specified order.



Data Read Back operations

Data Read Back is a data exchange channel for Power Shell to read the ASCII files produced by either user application programs or on-line data acquisition programs. Through the Data Read Back procedure, user ASCII files can be transferred and written to the diagram database data table of the Current Project, and the relevant data will be displayed on the power system diagram on the screen.

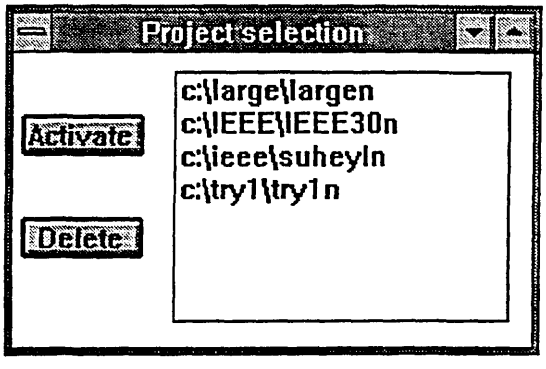


Anytime you want to display the latest data of the power system, i.e. the Current Project, running status on the diagram, you can always use Data Read Back to read the ASCII files produced by your on-line acquisition programs.

Part II Working with Files

Chapter 3 Managing projects

When you choose Project from the File menu in the Power Shell root window, the Project Selection dialog box will appear.



If there are some existing Power System Projects in Power Shell, you can activate or delete any of them.

Activating a Power System Project

- Select the name of the Power System Project from the list box, and then choose the Activate button.

Once a Power System Project has been activated, it will stay active until you open the Project Selection window again. An active Power System Project contains seven database tables and various data relationships. Except for the Create Project procedure, all the other procedures will not start until they have been notified of an active Power System Project by the Project Manager. To keep one Power System Project active is the pre-condition of the normal working status of Power Shell.

Deleting a Power System Project

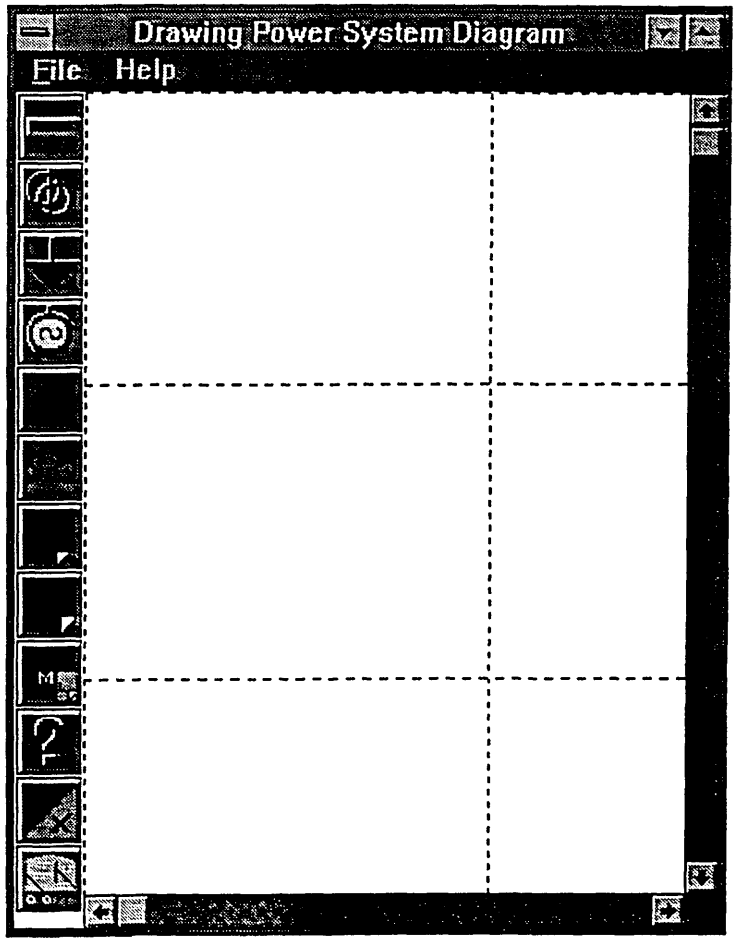
- Select the name of the Power System Project from the list box, and then choose the Delete button (Repeat this procedure if you want delete more than one Power System Project.).
- Close the Project Selection dialog box. From the control-menu box (upper left corner), choose the Close button.

When a Delete operation has been selected, the data file of the project is deleted only from the Project Manager register table but not physically from the hard disk. So there is still a chance for you to re-find the deleted Power System Project if you have deleted it by a mistake. You can use the Create Project option in the Database menu to re-find the Power System Project by typing in the name of the Power System Project that has been deleted. Since its data file has not been deleted from the hard disk, i.e. it still physically exists, the Create Project procedure need only rewrite this Power System Project to the Project Manager register table. The difference in creating a new Power System Project is that the corresponding database tables need to be created by the Create Project procedure.

When you decide to delete a Power System Project completely, i.e. delete its data file physically from the hard disk, you need use the File Manager facilities provided by MS-Windows.

Chapter 4 Drawing power system diagrams

For drawing a Power System diagram, you should choose Power System Diagram from the File menu in the Power Shell root window. If you do so, a Drawing Power System Diagram (DPSD) window will appear.



The DPSD working environment provides you with two DPSD menus, twelve drawing buttons and one three-layer drawing board. These three layers are: High-Layer of the Drawing Board(HLDB), Middle-Layer of the Drawing Board (MLDB) and Low-Layer of the Drawing Board(LLDB). Through DPSD you can draw, display, move, zoom in/out diagrams, and you can also access to data and update information.

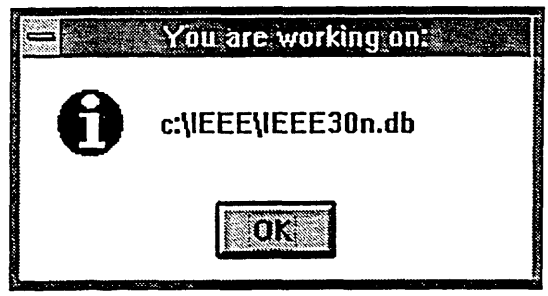
DPSD menus

There are two DPSD menus: File and Help. The File menu contains two options and the Help menu provides a brief introduction to the three layered drawing boards.

DPSD File menu

- **Open**

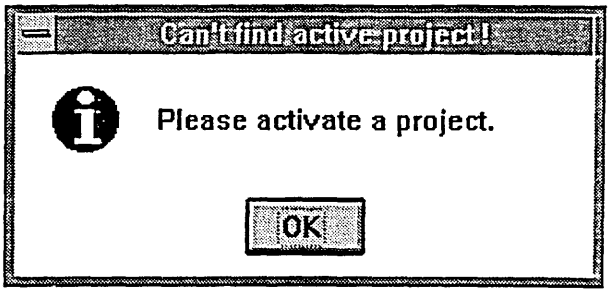
The DPSD operating target is assigned by the Project Manager. If you have activated a Power System Project through the Project option from the File menu in the Power Shell root window, then you can now use Open to register the activated Power System Project with the DPSD job scheduling table. If the Open procedure finds the Power System Project, an information dialog box confirming the target project will appear.



Now you can click the OK button to start the operation sequence.

In the operation if the diagram of the activated Power System Project exists, DPSD will display this diagram for you. Otherwise when a new Power System Project has been activated, a blank drawing board will be provided.

The Open procedure fails if no Power System Project has been activated. In this case an information dialog box requiring an active project will appear.



You should choose OK to close the information dialog box, and then close the DPSD window. Now choose Project from the File menu in the Power Shell root window to activate a Power System Project, then go through the DPSD procedure again.

- **Exit**

From the DPSD File menu, choose the Exit option. The DPSD window disappears.

Note
In the DPSD File menu, the option for closing a data table is not provided. Because multi-procedure concurrency is allowed in Power Shell, DPSD opens its data table for read and write only if any drawing procedure is being processed. DPSD will close the data table immediately after read and write to allow other procedures to use the data table. You do not need to use Open to open the data table repeatedly. For the registered Power System Project, DPSD will open and close the data table automatically according to your operation sequence.

DPSD buttons and their operations

Drawing busbars

- **How to draw a busbar**

 A Busbar button is provided for you to draw busbar symbols. You can draw a busbar at any position in the drawing board by, first, clicking the Busbar button. Now the mouse arrow has been changed to a busbar arrow  and

you can position the busbar arrow anywhere on the drawing board. When you fix a position for the busbar symbol, you must follow the steps below:

1. Paste- quickly press and release the mouse left button, then the busbar symbol is drawn on the drawing board.

2. Confirm - If you think the busbar is in the correct position, **do not** move the mouse, quickly press and release the mouse left button again. If you are not satisfied, you can delete the busbar from this position by **moving** the mouse and then quickly press and release the mouse left button again.

3. Input data- When you have confirmed the position of the busbar, an Input Device Parameter window containing a busbar data table will appear. Through this window you can input the busbar related data and parameters (see Input and update parameter in this Chapter).

Now the busbar drawing procedure is complete. If you want to draw more busbars then all you require is to repeat the above procedure starting with clicking the Busbar button.

- **Busbar colour**

When you obtain a busbar symbol from the busbar button, the colour of the busbar symbol depends on the current layer of the drawing board.

Working layer	Busbar colour
HLDB	blue
MLDB	red
LLDB	green

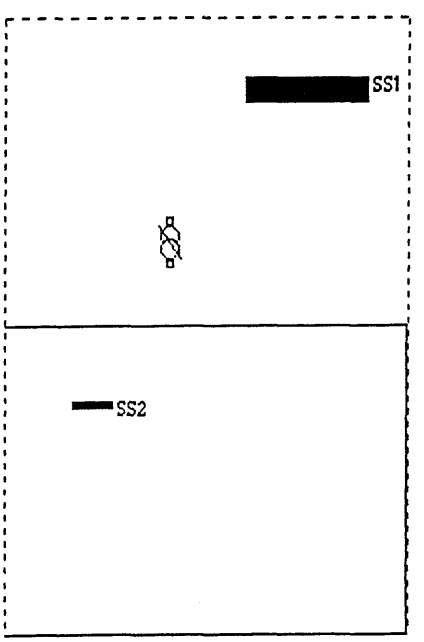
This means, for example, if you draw the High Voltage network at the HLDB then a blue busbars stands for High Voltage.

Drawing transformers

- **How to draw a transformer**

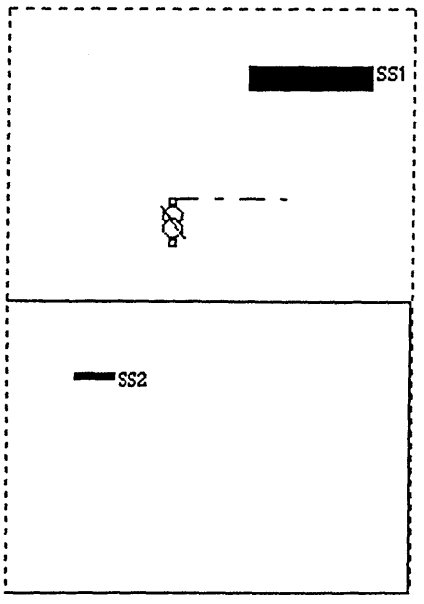
 A Transformer button is provided for you to draw transformer symbols. You can draw a transformer at any position in the drawing board by, first, clicking the transformer button. Now the mouse arrow has been changed to a transformer arrow  and you can position the transformer arrow anywhere on the drawing board. When you fix a position for the transformer symbol, you must follow the steps below:

1. Paste- quickly press and release the mouse left button, then the transformer symbol is drawn on the drawing board.



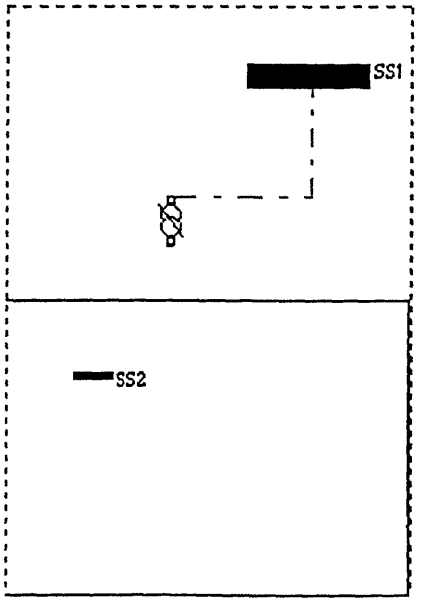
2. Top connecting line

a) When you have drawn a transformer symbol on the drawing board, a connecting line appears between the top of the transformer symbol and the transformer arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the top of the transformer symbol to the turning point will be drawn on the drawing board.



b)

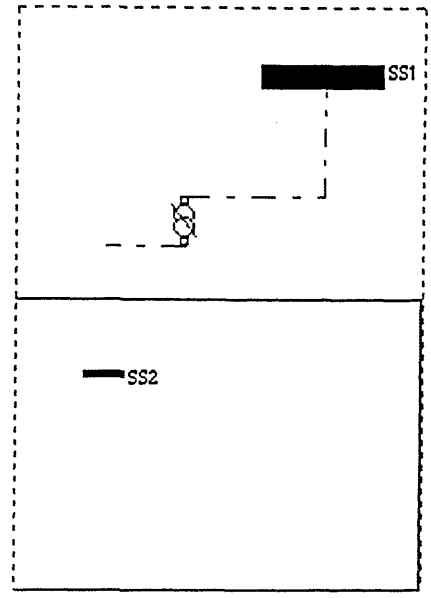
Pointing- When you finish the above step, a connecting line appears between the turning point and the transformer arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose a busbar that is to be connected to the transformer. Move the mouse to place the arrow on the target busbar.



Clicking- Point to the target busbar and quickly press and release the left mouse button, now a line from the top of the transformer symbol, through the turning point, to the busbar will be drawn on the drawing board.

3. Bottom connecting line

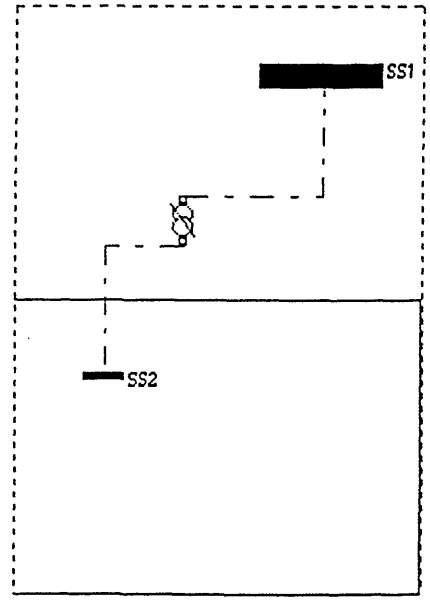
a) When you have drawn the top connecting line of a transformer symbol on the drawing board, a connecting line appears between the bottom of the transformer symbol and the transformer arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the bottom of the transformer symbol to the turning point will be drawn on the drawing board.



b)

Pointing- When you finish the above step, a connecting line appears between the turning point and the transformer arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose a busbar that is to be connected to the transformer. Move the mouse to place the arrow on the target busbar.

Clicking- Point to the target busbar and quickly press and release the left mouse button, now a line from the bottom of the transformer symbol, through the turning point, to the busbar will be drawn on the drawing board.



4. Confirm - If you think the transformer is in the correct position, **do not** move the mouse, quickly press and release the mouse left button again . If you are not satisfied, you can delete the transformer with the connecting lines from this position by **moving** the mouse and then quickly press and release the mouse left button again.

5. Input data- When you have confirmed the position of the transformer, an Input Device Parameter (IDP) window containing transformer data table appears. Through this window you can input the transformer related data and parameters (see Input and update parameter in this Chapter).

6. Define input and output busbars- After you input data to the transformer you need to define its input and output busbars. Use the Query button to query the transformer you just drew (see Querying graphical element in this Chapter for details of using Query button), the names of two busbars connected to this transformer will be displayed in the IDP window. You must use 'I' to define one of them as the input busbar of the transformer, and use 'O' to define the another as the output busbar (I/O item is provided by the IDP window, see Input and update parameter in this Chapter).

Now the transformer drawing procedure is complete. If you want to draw more transformers then all you require is to repeat the above procedure starting with clicking the Transformer button.

- **What kind of busbars will respond to the connecting operation**

If you are working only on one layer of the drawing board, then any busbar will respond to the connection.

If you are working on the overlapped layers of the drawing board, the busbars on the current working layer and which on the OFSB of the layer(s) over the current layer will respond to the connection.

Working layer	Busbars that will respond to the connection
HLDB	Busbars on the HLDB
MLDB	Busbars on the MLDB Busbars on the OFSB of the HLDB
LLDB	Busbars on the LLDB Busbars on the OFSB of the HLDB Busbars on the OFSB of the MLDB

The elements on the OFSB(s) of the layer(s) over the current working layer will show two different colours.

	OFSB	Other sub board
HLDB element colour	blue	dark blue
MLDB element colour	red	dark red
LLDB element colour	green	
	Will respond to the connection	

If the busbar you want to connect is not on the OFSB (Operation Focus Sub Board, see High/Middle/Low-Layer of the Drawing Board in this Chapter) you must first change the OFSB, i.e. to make the sub board containing this busbar an OFSB. Then when the busbar responds by changing its colour you can start connecting operation.

- **Transformer colour**

When you obtain a transformer symbol from the Transformer button, the colour of the transformer symbol depends on the current layer of the drawing board.

Working layer	Transformer colour
HLDB	blue
MLDB	red
LLDB	green

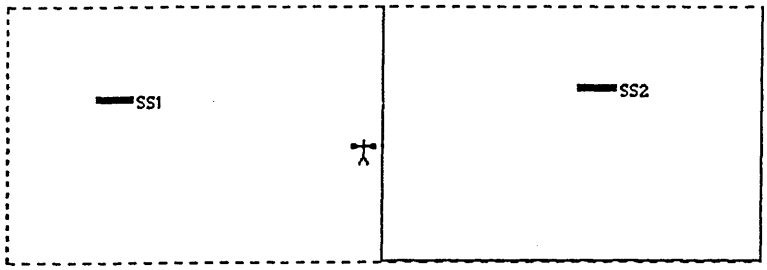
This means, for example, if you draw the High Voltage network at the HLDB then a blue transformer stands for High Voltage.

Drawing cables (lines)

- **How to draw a cable (line)**

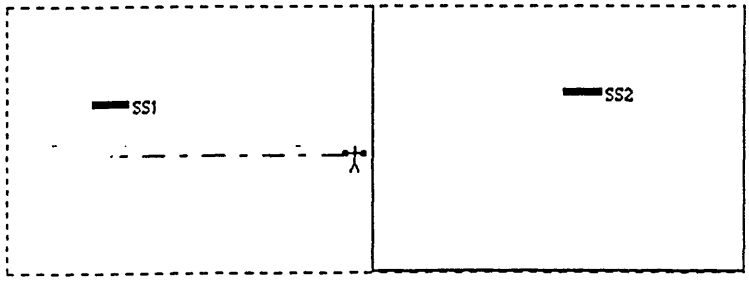
 A Cable button is provided for you to draw cable symbols. You can draw a cable at any position in the drawing board by, first, clicking the Cable button. Now the mouse arrow has been changed to a cable arrow  and you can position the cable arrow anywhere on the drawing board. When you fix a position for the cable symbol, you must follow the steps below:

1. Paste- quickly press and release the mouse left button, then the cable symbol is drawn on the drawing board.



2. Left connecting line

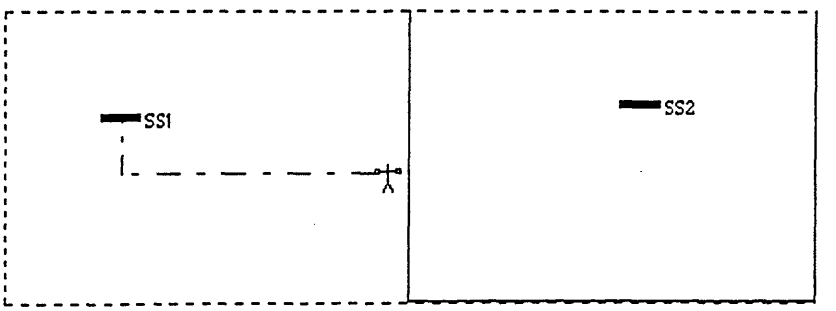
   a) When you have drawn a cable symbol on the drawing board, a connecting line appears between the left of the cable symbol and the cable arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the left of the cable symbol to the turning point will be drawn on the drawing board.



b)

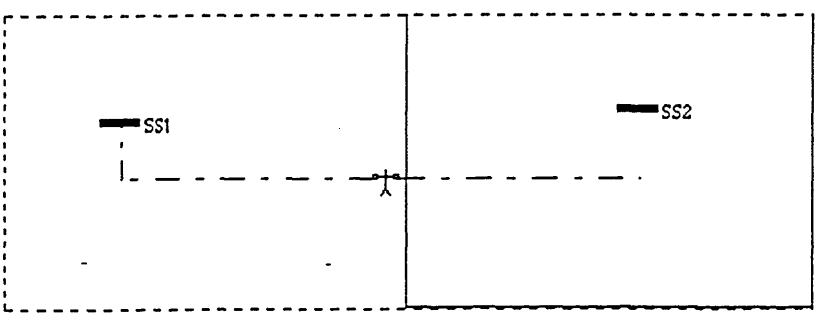
Pointing- When you finish the above step, a connecting line appears between the turning point and the cable arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose a busbar that is to be connected to the cable. Move the mouse to place the arrow on the target busbar.

Clicking- Point to the target busbar and quickly press and release the left mouse button, now a line from the left of the cable symbol, through the turning point, to the busbar will be drawn on the drawing board.



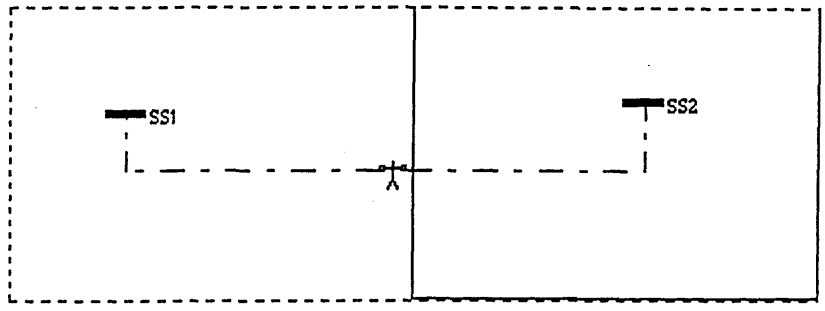
3. Right connecting line

a) When you have drawn the left connecting line of a cable on the drawing board, a connecting line appears between the right of the cable symbol and the cable arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the right of the cable symbol to the turning point will be drawn on the drawing board.



b)

Pointing- When you finish the above step, a connecting line appears between the turning point and the cable arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose a busbar that is to be connected to the cable. Move the mouse to place the arrow on the target busbar.



Clicking- Point to the target busbar and quickly press and release the left mouse button, now a line from the right of the cable symbol, through the turning point, to the busbar will be drawn on the drawing board.

4. Confirm- If you think the cable is in the correct position, **do not** move the mouse, quickly press and release the mouse left button again. If you are not satisfied, you can delete the cable with the connecting lines from this position by **moving** the mouse and then quickly press and release the mouse left button again.

5. Input data- When you have confirmed the position of the cable, an Input Device Parameter (IDP) window containing a cable data table appears. Through this window you can input the cable related data and parameters (see Input and update parameter in this Chapter).

6. Define input and output busbars - After you input data to the cable you need to define its input and output busbars. Use the Query button to query the cable you just drew (see Querying graphical element in this Chapter for details of using Query button), the names of two busbars connected to this cable will be displayed in the IDP window. You must use 'I' to define one of them as the input busbar of the cable, and use 'O' to define the another as the output busbar (I/O item is provided by the IDP window, see Input and update parameter in this Chapter).

Now the cable drawing procedure is complete. If you want to draw more cables then all you require is to repeat the above procedure starting with clicking the Cable button.

- **What kind of busbars will respond to the connecting operation**

If you are working only on one layer of the drawing board, then any busbar will respond to the connection.

If you are working on the overlapped layers of the drawing board, the busbars on the current working layer and which on the OFSB(s) of the layer(s) over the current layer will respond to the connection.

Working layer	Busbars that will respond to the connection
HLDB	Busbars on the HLDB
MLDB	Busbars on the MLDB Busbars on the OFSB of the HLDB
LLDB	Busbars on the LLDB Busbars on the OFSB of the HLDB Busbars on the OFSB of the MLDB

The elements on the OFSB(s) of the layer(s) over the current working layer will show two different colours.

	OFSB	Other sub board
HLDB element colour	blue	dark blue
MLDB element colour	red	dark red
LLDB element colour	green	
	Will respond to the connection	

If the busbar you want to connect is not on the OFSB (Operation Focus Sub Board, see High/Middle/Low-Layer of the Drawing Board in this Chapter) you must first change the OFSB, i.e. to make the sub board containing this busbar an OFSB. Then when the busbar responds by changing its colour you can start the connecting operation.

- **Cable colour**

When you obtain a cable symbol from the Cable button, the colour of the cable symbol depends on the current layer of the drawing board.

Working layer	Cable colour
HLDB	blue
MLDB	red
LLDB	green

This means, for example, if you draw the High Voltage network at the HLDB then a blue cable stands for High Voltage.

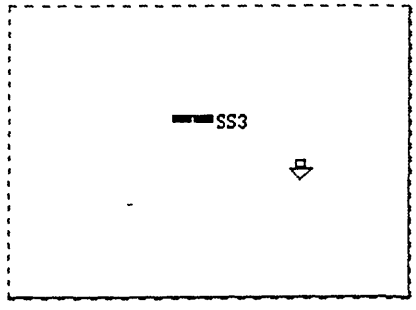
Drawing loads

- **How to draw a lord**

 A Load button is provided for you to draw load symbols. You can draw a load at any position in the drawing board by, first, clicking the load button.

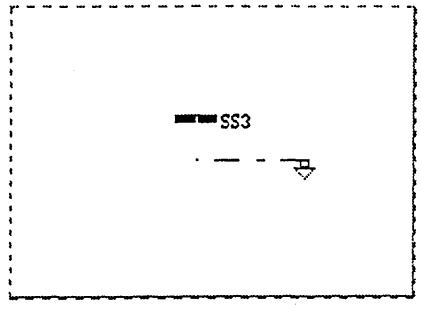
Now the mouse arrow has been changed to a load arrow  and you can position the load arrow anywhere on the drawing board. When you fix a position for the load symbol, you must follow the steps below:

1. Paste- quickly press and release the mouse left button, then the load symbol is drawn on the drawing board.



2. Top connecting line

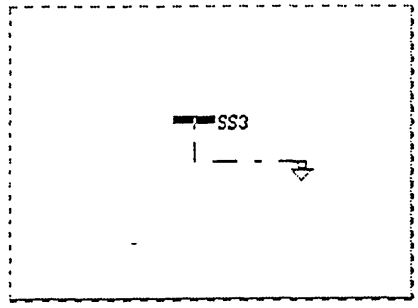
a) When you have drawn a load symbol on the drawing board, a connecting line appears between the top of the load symbol and the load arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the top of the load symbol to the turning point will be drawn on the drawing board.



b)

Pointing- When you finish the above step, a connecting line appears between the turning point and the load arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose a busbar that is to be connected to the load. Move the mouse to place the arrow on the target busbar.

Clicking- Point to the target busbar and quickly press and release the left mouse button, now a line from the top of the load symbol, through the turning point, to the busbar will be drawn on the drawing board.



3. Confirm- If you think the load is in the correct position, **do not** move the mouse, quickly press and release the mouse left button again . If you are not satisfied, you can delete the load with the connection line from this position by **moving** the mouse and then quickly press and release the mouse left button again.
4. Input data- When you have confirmed the position of the load, an Input Device Parameter window containing a load data table appears. Through this window you can input the load related data and parameters (see Input and update parameter in this Chapter).

Now the load drawing procedure is complete. If you want to draw more loads then all you require is to repeat the above procedure starting with clicking the Load button.

- **What kind of busbars will respond to the connecting operation**

If you are working only on one layer of the drawing board, then any busbar will respond to the connection.

If you are working on the overlapped layers of the drawing board, the busbars on the current working layer and which on the OFSB(s) of the layer(s) over the current layer will respond to the connection.

Working layer	Busbars that will respond to the connection
HLDB	Busbars on the HLDB
MLDB	Busbars on the MLDB Busbars on the OFSB of the HLDB
LLDB	Busbars on the LLDB Busbars on the OFSB of the HLDB Busbars on the OFSB of the MLDB

The elements on the OFSB(s) of the layer(s) over the current working layer will show two different colours.

	OFSB	Other sub board
HLDB element colour MLDB element colour LLDB element colour	blue red green	dark blue dark red
	Will respond to the connection	

If the busbar you want to connect is not on the OFSB (Operation Focus Sub Board, see High/Middle/Low-Layer of the Drawing Board in this Chapter) you must first change the OFSB, i.e. to make the sub board containing this busbar an OFSB. Then when the busbar responds by changing its colour you can start the connecting operation.

- **Load colour**

When you obtain a load symbol from the Load button, the colour of the load symbol depends on the current layer of the drawing board.

Working layer	Load colour
HLDB	blue
MLDB	red
LLDB	green

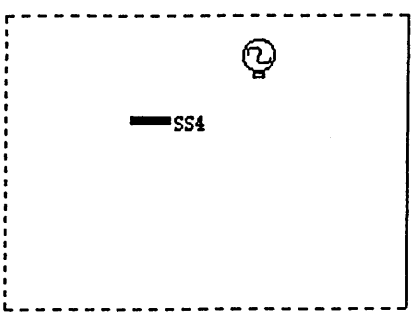
This means, for example, if you draw the High Voltage network at the HLDB then a blue load stands for High Voltage.

Drawing generators

- **How to draw a generator**

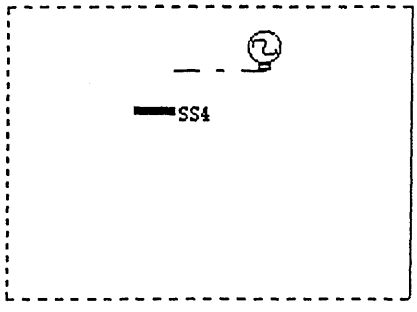
 A Generator button is provided for you to draw generator symbols. You can draw a generator at any position in the drawing board by, first, clicking the generator button. Now the mouse arrow has been changed to a generator arrow  and you can position the generator arrow anywhere on the drawing board. When you fix a position for the generator symbol, you must follow the steps below:

1. Paste- quickly press and release the mouse left button, then the Generator symbol is drawn on the drawing board.



2. Bottom connecting line

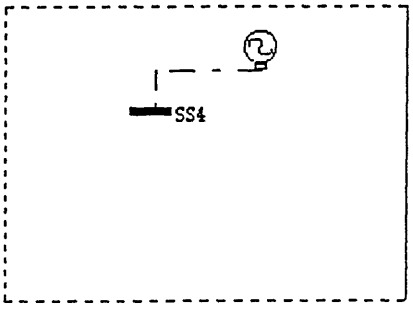
   a) When you have drawn a generator symbol on the drawing board, a connecting line appears between the bottom of the generator symbol and the generator arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the bottom of the generator symbol to the turning point will be drawn on the drawing board.



b)

Pointing- When you finish the above step, a connecting line appears between the turning point and the generator arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose a busbar that is to be connected to the generator. Move the mouse to place the arrow on the target busbar.

Clicking- Point to the target busbar and quickly press and release the left mouse button, now a line from the bottom of the generator symbol, through the turning point, to the busbar will be drawn on the drawing board.



3. Confirm- If you think the generator is in the correct position, **do not** move the mouse, quickly press and release the mouse left button again . If you are not satisfied, you can delete the generator with the connecting line from this position by **moving** the mouse and then quickly press and release the mouse left button again.

4. Input data- When you have confirmed the position of the generator, an Input Device Parameter window containing a generator data table appears. Through this window you can input the generator related data and parameters (see Input and update parameter in this Chapter).

Now the generator drawing procedure is complete. If you want to draw more generators then all you require is to repeat the above procedure starting with clicking the Generator button.

- **What kind of busbars will respond to the connecting operation**

If you are working only on one layer of the drawing board, then any busbar will respond to the connection.

If you are working on the overlapped layers of the drawing board, the busbar on the current working layer and which on the OFSB(s) of the layer(s) over the current layer will respond to the connection.

Working layer	Busbars that will respond to the connection
HLDB	Busbars on the HLDB
MLDB	Busbars on the MLDB Busbars on the OFSB of the HLDB
LLDB	Busbars on the LLDB Busbars on the OFSB of the HLDB Busbars on the OFSB of the MLDB

The elements on the OFSB(s) of the layer(s) over the current working layer will show two different colours.

	OFSB	Other sub board
HLDB element colour	blue	dark blue
MLDB element colour	red	dark red
LLDB element colour	green	
	Will respond to the connection	

If the busbar you want to connect is not on the OFSB (Operation Focus Sub Board, see High/Middle/Low-Layer of the Drawing Board in this Chapter) you must first change the OFSB, i.e. to make the sub board containing this busbar an OFSB. Then when the busbar responds by changing its colour you can start the connecting operation.

- **Generator colour**

When you obtain a generator symbol from the Generator button, the colour of the generator symbol depends on the current layer of the drawing board.

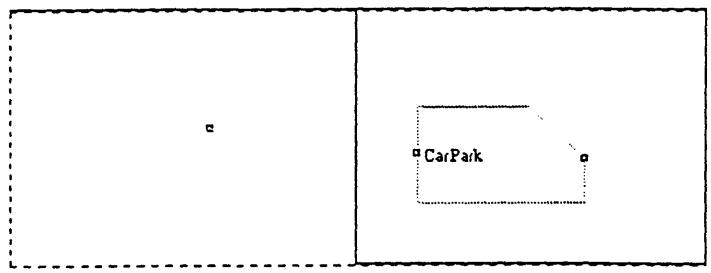
Working layer	Generator colour
HLDB	blue
MLDB	red
LLDB	green

This means, for example, if you draw the High Voltage network at the HLDB then a blue generator stands for High Voltage.

Drawing Map-line

 A Map-line button is provided for you to draw map-line symbols. You can draw a map-line at any position in the drawing board by, first, clicking the map-line button. Now the mouse arrow has been changed to a map-line arrow  and you can position the map-line arrow anywhere on the drawing board. When you fix a position for the map-line symbol, you must follow the steps below:

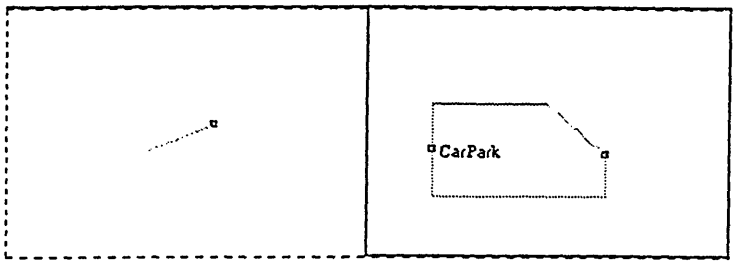
1. Paste- quickly press and release the mouse left button, then the map-line symbol is drawn on the drawing board.



2. Left conneeting line

a) When you have drawn a Map-line symbol on the drawing board, a connecting line appears between the left of the map-line symbol and the map-line arrow. The direction and length of the line will be changed as the arrow

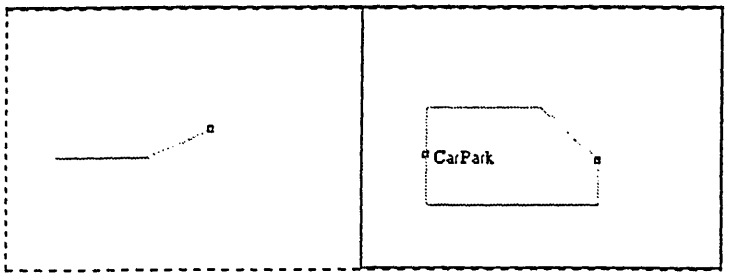
moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the left of the map-line symbol to the turning point will be drawn on the drawing board.



b)

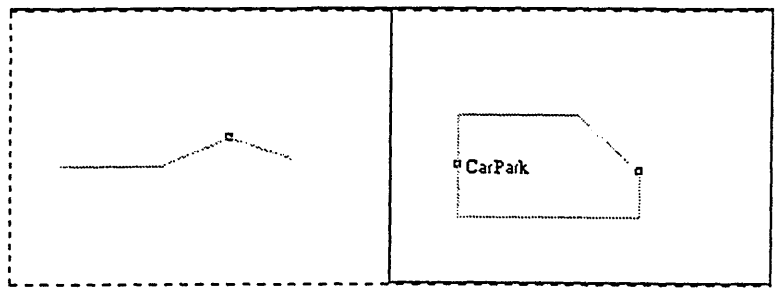
Pointing- When you finish the above step, a map line appears between the turning point and the map-line arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose an end point of the map-line. Move the mouse to place the arrow on the target point.

Clicking- Point to the target point and quickly press and release the left mouse button, now a line from the left of the map-line symbol, through the turning point, to the end point will be drawn on the drawing board.



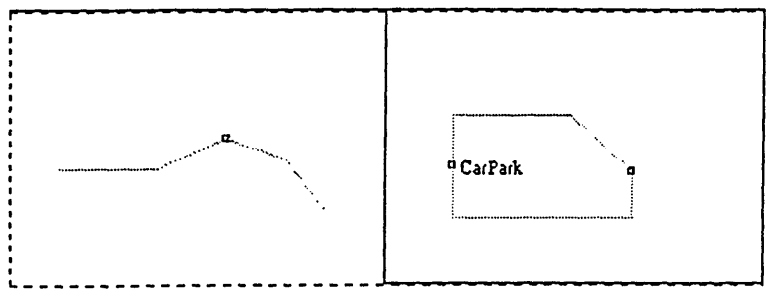
3. Right connecting line

a) When you have drawn the left connecting line of a map-line symbol on the drawing board, a connecting line appears between the right of the map-line symbol and the map-line arrow. The direction and length of the line will be changed as the arrow moves. When you decide the first turning point, quickly press and release the left mouse button, then a line from the right of the map-line symbol to the turning point will drawn on the drawing board.



b)

Pointing- When you finish the above step, a map line will appear between the turning point and the map-line arrow. The direction and length of the line will be changed as the arrow moves. Now you should choose an end point of the map-line. Move the mouse to place the arrow on the target point.



4. Confirm- If you think the map line is in the correct position, **do not** move the mouse, quickly press and release the mouse left button again. If you are not satisfied, you can delete the map line by **moving** the mouse and then quickly press and release the mouse left button again.

5. Input data- When you have confirmed the position of the map line, an Input Device Parameter window containing a map-line data table appears. Through this window you can input the name of the map-line related place (see Input and update parameter in this Chapter).

Now the map line drawing procedure is complete. If you want to draw more map-lines then all you require is to repeat the above procedure starting with clicking the Map-line button.

High-Layer of the Drawing Board

 The High-Layer of the Drawing Board (HLDB) is a drawing board containing sub drawing boards formed by crossed blue lines. The number of these sub drawing boards is n×m; where n=1111, m=1111. You are allowed to draw up to 8 elements in each sub drawing board. So you can draw 8×1111^2 elements on the HLDB if your hard disk has a high enough capacity. When your enter the Drawing Power System Diagram (DPSD) window, you are now at the HLDB central area.

You can move the HLDB up, down, left and right through the scroll bar provided. Among these sub drawing boards, one is the Current Operation Focus Sub Board (OFSB), which is distinguished by a solid blue line border. An OFSB is set up automatically according to the current operation. When you put the mouse within a sub drawing board area and click the mouse left button, then this sub board becomes the OFSB.

If you wish your network diagram well-structured you may draw the network of High/Middle/Low Voltage level on the HLDB, MLDB (see Middle-Layer of the Drawing Board section) and LLDB (see Low-Layer of the Drawing Board section) respectively. The elements on the different layers, i.e. HLDB, MLDB and LLDB, can be connected. If you wish your network diagram to be produced with geographical information, you may draw the geographical map on the HLDB and draw the power network diagram on the MLDB and the LLDB. HLDB, MLDB and LLDB are transparent, and can be overlapped in the display window to give an overview of the power system diagram integrating the diagrams you have drawn on different layers.

If you are currently working on the MLDB and want to switch the drawing board to the HLDB, just click the HLDB button  DPSD will switch, i.e. Zoom Out, from the OFSB of the MLDB to the HLDB. The following switches between the HLDB and the MLDB are available:

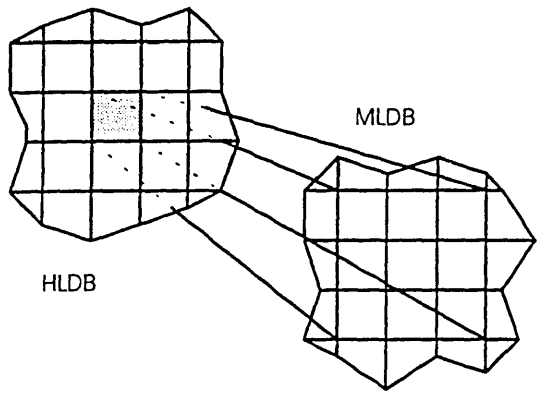
Current board	to	Target board
FOSB of HLDB	Zoom In	MLDB
FOSB of MLDB	Zoom Out	HLDB

Middle-Layer of the Drawing Board

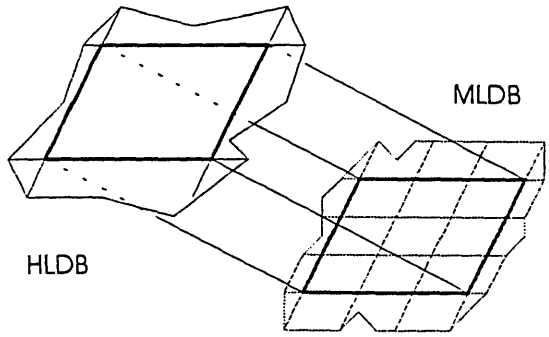
The Middle-Layer of the Drawing Board (MLDB) is a drawing board containing sub drawing boards formed by crossed red lines. The number of these sub drawing boards is n×m; where n=3333, m=3333. You are allowed to draw up to 8 elements in each sub drawing board, so you can draw 8×3333^2 elements on the MLDB if the hard disk has a high enough capacity.

You can move the MLDB up, down, left and right through the scroll bar provided. Among these sub drawing boards, one is the Current Operation Focus Sub board (OFSB), which is distinguished by a solid red line border. An OFSB is set up automatically according to the current operation. When you put the mouse within a sub drawing board area and click the mouse left button, then this sub board becomes the OFSB.

There is a projecting relationships between the MLDB and the HLDB. The projecting proportion from the HLDB to the MLDB is 1:3.



Because the number of power network elements at the Middle Voltage Level is much larger than that at the High Voltage Level, there is more drawing space on the MLDB provided by the Drawing Power System Diagram (DPSD) window. You can Zoom In from the FOSB of the HLDB to the MLDB.



If you wish your network diagram well-structured you may draw the network of High/Middle/Low Voltage level on the HLDB (see High-Layer of the Drawing Board section), MLDB and LLDB (see Low-Layer of the Drawing Board section) respectively. The elements on the different layers, i.e. HLDB, MLDB and LLDB, can be connected. If you wish your network diagram to be produced with geographical information, you may draw the geographical map on the HLDB and draw the power network diagram on the MLDB and the LLDB. HLDB, MLDB and LLDB are transparent, and can be overlapped in the display window to give an overview of the power system diagram integrating the diagrams you have drawn on different layers.

If you are currently working on the HLDB and want to switch the drawing board to the MLDB, just click the MLDB button  . DPSD will automatically switch, i.e. Zoom In, from the OFSB of the HLDB to the projectively related area on the MLDB. If you are currently working on the LLDB and want to switch the drawing board to the MLDB, just click the MLDB button. DPSD will automatically switch, i.e. Zoom Out, from the OFSB of the LLDB to the projectively related area on the MLDB. The follow switches between different boards are available:

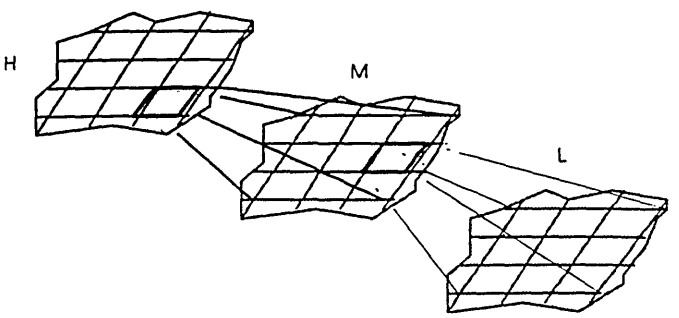
Current board	to	Target board
FOSB of HLDB	Zoom In	MLDB
FOSB of MLDB	Zoom Out	HLDB
FOSB of MLDB	Zoom In	LLDB
FOSB of LLDB	Zoom Out	MLDB

Low-Layer of the Drawing Board

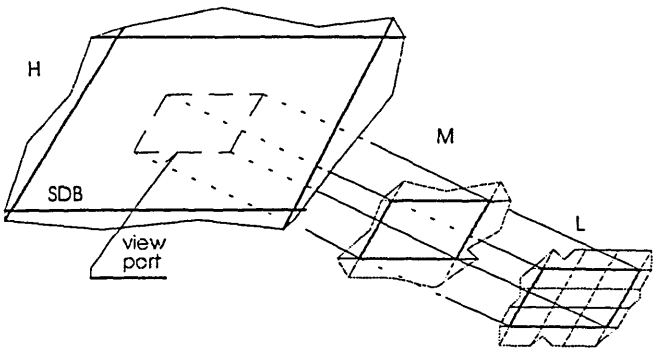
The Low-Layer of the Drawing Board (LLDB) is a drawing board containing sub drawing boards formed by crossed green lines. The number of these sub drawing boards is n×m; where n=9999, m=9999. You are allowed to draw up to 8 elements in each sub drawing board. Therefore you can draw 8×9999^2 elements on the LLDB if the hard disk has a high enough capacity.

You can move the LLDB up, down, left and right through the scroll bar provided. Among these sub drawing boards, one is the Current Operation Focus Sub Board (OFSB), which is distinguished by a solid green line border. An OFSB is set up automatically according to the current operation. When you put the mouse within a sub drawing board area and click the mouse left button, then this sub board becomes the OFSB.

There is a projecting relationships between the LLDB and the MLDB. The projecting proportion from the MLDB to the LLDB is 1:3.



Because the number of power network elements at the Low Voltage level is much larger than that at the Middle Voltage level, there is more drawing space on the LLDB provided by the Drawing Power System Diagram window. You can Zoom In from the FOSB of the MLDB to the LLDB.



The LLDB has projective relationships with the HLDB and the MLDB. Since

L-element number >> M-element number >> H-element number,

where L/M/H-element number stands for the number of power system elements at Low/ Middle/High Voltage level, DPSD provides you more drawing space on the LLDB then that on the MLDB.

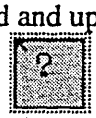
If you wish your network diagram well-structured you may draw the network of High/Middle/Low Voltage level on the HLDB (see High-Layer of the Drawing Board section), MLDB (see Middle-Layer of the Drawing Board section) and LLDB respectively. The elements on the different layers, i.e. HLDB, MLDB and LLDB, can be connected. If you wish your network diagram to be produced with geographical information, you may draw the geographical map on the HLDB and draw the power network diagram on the MLDB and the LLDB. HLDB, MLDB and LLDB are transparent, and can be overlapped in the display window to give an overview of the power system diagram integrating the diagrams you have drawn on different layers.

If you are currently working on the MLDB and want to switch the drawing board to the LLDB, just click the LLDB button  . DPSD will automatically switch, i.e. Zoom In, from the OFSB of the MLDB to the projectively related area on the LLDB. The follow switches between the LLDB and the MLDB are available:

Current board	to	Target board
FOSB of MLDB	Zoom In	LLDB
FOSB of LLDB	Zoom Out	MLDB

Querying graphical element

- **How to query element**

 Using the Query button you can query any element of the power system for read and update relative data. When you click the Query button, a query arrow  appears. Using the arrow click an element of the power system on the drawing board, an Input Device Parameter (IDP) window (see Input and update parameter in this Chapter later) for the clicked element appears.

Through the IDP you can read or update data relative to the element. If you have updated the data, you should click OK from the IPU File menu before you close the window. If you have only read the data, you should click Cancel from IPU File menu before you close the window.

- **What kind of elements will respond to your query**

If you are working only on one layer of the drawing board, any elements of the power system can respond your query.

If you are working on the overlapped layers of the drawing board, the elements on the current working layer and which on the OFSB(s) of the layer(s) over the current layer will respond to your query.

Working layer	Elements that will respond to your query
HLDB	Elements on the HLDB
MLDB	Elements on the MLDB Elements on the OFSB of the HLDB
LLDB	Elements on the LLDB Elements on the OFSB of the HLDB Elements on the OFSB of the MLDB

The elements on the OFSB(s) of the layer(s) over the current working layer will show two different colours.

	OFSB	Other sub board
HLDB element colour	blue	dark blue
MLDB element colour	red	dark red
LLDB element colour	green	
	Will respond to the query	

If the element you want to query is not on the OFSB, you must first change the OFSB, i.e. make the sub board containing this element an OFSB (see High/ Middle/Low-Layer of the Drawing Board). Then when the element responds by changing its colour, you can start query operation.

Information Cut

A Mark button is provided for information cut in the data exchange process. If you want to transfer part information of Power System Project to the ASCII files through the Data Export facilities provided by Power Shell, you should use the Mark button to mark the elements whose data you want to transfer.

In the Data Export operation, you have two options: Part of Network and Whole Network. If you choose the Part of Network option, the Data Export procedure will check and transfer the data of the marked elements of the Current Project to ASCII files. Those unmarked elements will be ignored.

- **Marking elements**

When you click the Mark Button, a mark arrow appears. Using the arrow you can mark or unmark the elements of the Current Project. If you use the arrow to click an unmarked element, this element now has been marked and its colour changes. If you use the arrow to click an marked element then this element now is unmarked and its colour changes back to normal.

- **What kind of elements will respond to the mark operation**

If you are working only on one layer of the drawing board, any element of the power system can respond your marking.

If you are working on the overlapped layers of the drawing board, the elements on the current working layer and which on the OFSB(s) of the layer(s) over the current layer will respond to your marking.

Working layer	Elements that will respond to your marking
HLDB	Elements on the HLDB
MLDB	Elements on the MLDB Elements on the OFSB of the HLDB
LLDB	Elements on the LLDB Elements on the OFSB of the HLDB Elements on the OFSB of the MLDB

The elements on the OFSB(s) of the layer(s) over the current working layer will show two different colours.

	OFSB	Other sub board
HLDB element colour	blue	dark blue
MLDB element colour	red	dark red
LLDB element colour	green	
	Will respond to the query	

If the element you want to query is not on the OFSB, you must first change the OFSB, i.e. make the sub board contains this element an OFSB (see High/Middle/ Low-Layer of the Drawing Board). Then when the element responds by changing its colour, you can start mark operation.

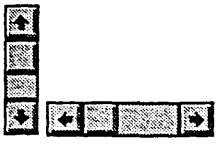
Diagram Only operations

 A Diagram Only button is provided for temporarily hiding the data displayed on the power system diagram.

The Data Read Back procedure is for reading back the ASCII files produced by user application programs to the database table of the Current Project. The data obtained by Data Read Back can be displayed on the power system diagram. If you would like this data not to be displayed in some circumstances, click the Diagram Only button, the data will be invisible temporarily. But the data is not deleted from the database, when you click any button the data will be again displayed. If you wish a higher speed when moving the power system diagram,

you may first hide the data through the Diagram Only button, then start the Move operation.

Scroll bar

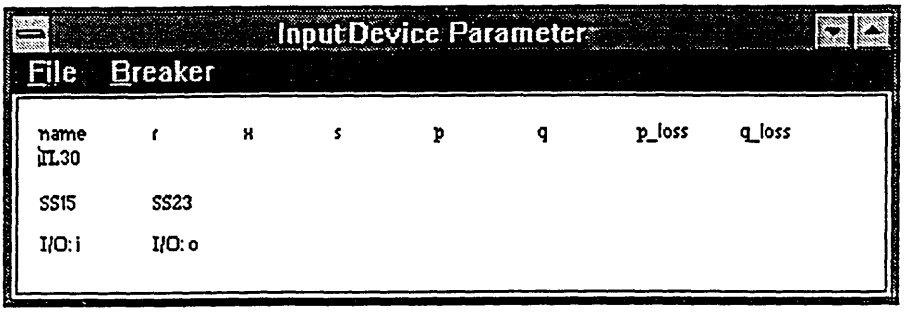


Power Shell provides a three-layer and size-unlimited drawing board. When the area covering the power system diagram is larger than the display window, out side part of the diagram can be viewed by using the scroll bar to move the drawing board.

If you are working on overlapped layers of the drawing board, then the drawing boards will be moved synchronously.

Input and update parameter

When you draw a new element or query an existing element on the drawing board, the Input Device Parameter (IDP) window appears.



- **For a new element**

For a new element, IDP provides you with a blank data table. You have the following options.

1. Input data

To input data for a new element, you should

a) Type data to the IDP data table;

Note: In Power Shell the name of the Power System Project elements must be unique, i.e. two different elements cannot have the same name.

b) Choose OK from the IDP File menu;

c) Choose Close from the IDP control-menu box  (upper left corner of the IDP window).

Now IDP is closed. The data you input has been recorded in the Current Project database tables, and the name of the new element will appear on the power system diagram of the Current Project.

2. Delete an element

To delete a new element, you should

a) Choose Cancel from the IDP File menu;

b) Choose Close from the IDP control-menu box  (upper left corner of the IDP window).

Now IDP is closed and the new element has been deleted from the drawing board.

- **For a data-existing element**

For a data-existing element, IDP provides the data table for you . You have the following options.

1. Read data

To read data, you should

a) Read data from the given data table;
b) Choose Cancel from the IDP File menu;

c) Choose Close from the IDP control-menu box  (upper left corner of the IDP window).

2. Update data

To update element data, you should

a) Change the parameters in the data table by using the keyboard;
b) Change the breaker status by choosing On or Off from the IDP Breaker menu;
c) Choose OK from the IDP File menu;
d) Choose Close from the IDP control-menu box  (upper left corner of the IDP window).

Now the IDP is closed; and the data has been updated in the database table.

3. Delete an element

To delete a data-existing element from the Current Project database tables, you should

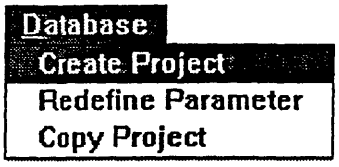
a) Choose Delete from the IDP File menu;
b) Choose Close from the IDP control-menu box  (upper left corner of the IDP window).

Now the IDP is closed; and all the data for this element has been deleted from the database tables

Note: To maintain the correct relationships between data in the database tables, before deleting a busbar you must delete all the elements that are directly connected to the busbar, e.g. Cable, Load, Generator and Transformer. Then you can delete this busbar. Otherwise an error of ill data relationships may occur.

Part III Working with Databases

The Database menu in the Power Shell root window is for managing Power System Project data.



Through the Create Project option you can create a new Power System Project containing 7 database tables and 7 index files. When a new Power System Project has been created, it is automatically registered in the Project Manager table. Then you can activate the new Power System Project and input data to it through the options in the File menu (see Part II Working with Files). You can also redefine the data structure of the parameter table for the following power system elements through the Redefine Parameter option.

- Busbar
- Cable(Line)
- Transformer
- Generator
- Load

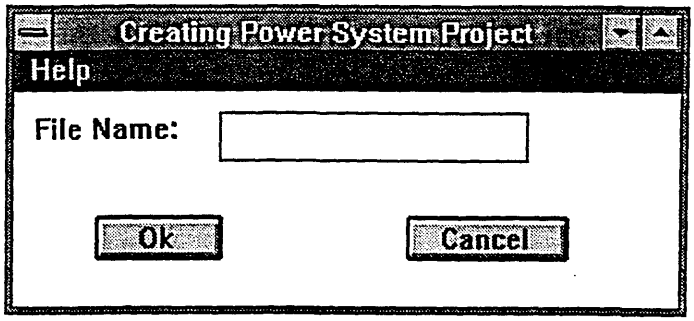
If you want to use an existing diagram and the relative data when you create a new Power System Project, you can choose the Copy Project option to copy the diagram and the data. The following Chapters in this Part detail the procedures of Creating projects, Redefining parameters and Copying project data respectively.

Chapter 5 Creating projects

To create a Power System Project for your own power system network is an essential step if you want to work with it later in Power Shell.

Creating a Power System Project

From the Database menu in the Power Shell root window, choose Create Project, a Creating Power System Project (CPSP) window appears.



Now you should follow the steps below to create a Power System Project.

Input project name

The first thing you should do is to input a legal name for the Power System Project in the File Name box. A legal name of a Power System Project should take the follow pattern.

DriveName:\PathName\ProjectName

where
DriveName should be either C or D;
PathName should be composed of up to 8 ASCII characters;
ProjectName should be composed of up to 6 ASCII characters.
The extension name is not necessary.

After you input a correct Power System Project name, you should click the OK button. Then Create Project will create the relevant database tables for your Power System Project under the PathName directory.

Cancel creating process

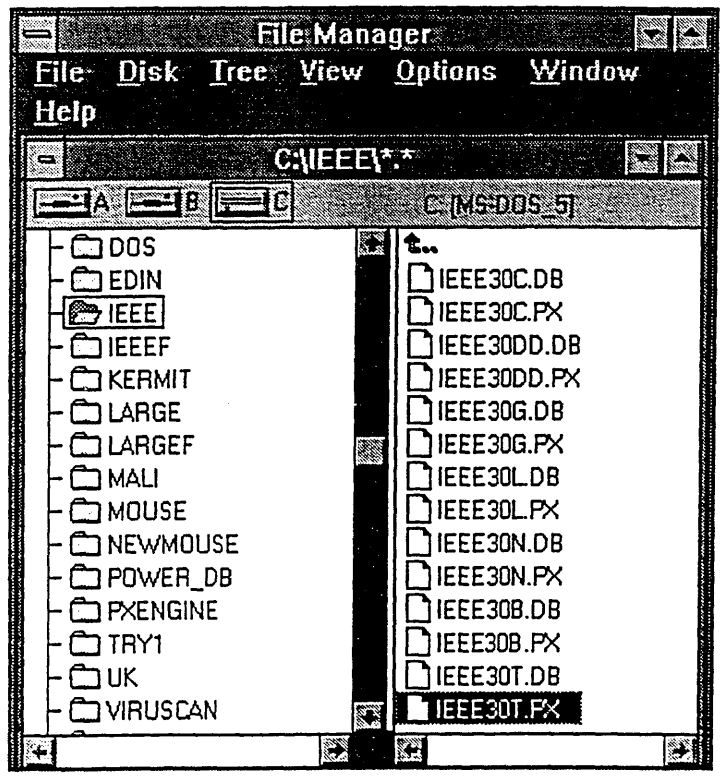
If you do not want to continue the creating process after your open the CPSP, you can use the Cancel button to close and exit CPSP.

Producing data table names

After the Create Project procedure, a new Power System Project will be automatically registered in the Project Manger table. The names of the seven data base tables produced by the Create Project program have been added with extension names according to the following rules

ProjectNameN.db - Network configuration table
ProjectNameDD.db - Drawing diagram table
ProjectNameB.db - Busbar parameter table
ProjectNameL.db - Load parameter table
ProjectNameT.db - Transformer parameter table
ProjectNameG.db - Generator parameter table
ProjectNameC.db - Cable(Line) parameter table

Similarly seven index files have been generated in the same directory.

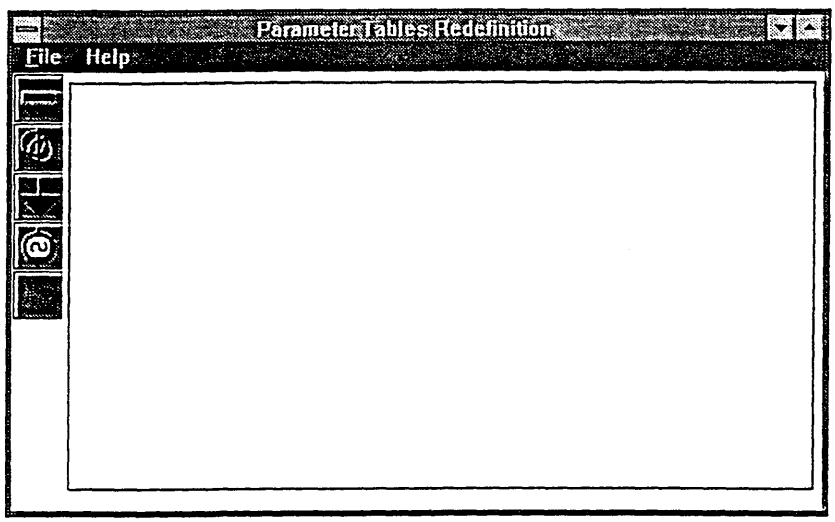


For a newly created Power System Project, the data structure of the parameter table in the data base has been defined in a default structure. If your application program requires a different data structure, you can redefine the data structure of the parameter database table (see Chapter 6 Redefining parameter).

Chapter 6 Redefining parameters

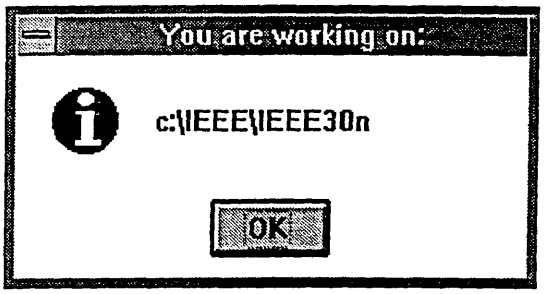
As you create a new Power System Project, five parameter database tables for the power system elements are produced by Power Shell in the default structures. If you need a different data structure for your application program, you can redefine the data structure for any of these parameter tables through Redefine Parameter. When the structure of a parameter table has been redefined, a new parameter table will be produced in the redefined structure to replace the old one. The name of this table will not be changed. The new table belongs to the same Power System Project. When the new table has been produced, the data in the old one is ignored. So the effective way in case of redefinition is to redefine the data structure before you put any data in the default table after you create a new Power System Project.

When you choose the Redefine Parameter option from the Database menu in the Power Shell root window, a Parameter Table Redefinition (PTR) window for the Current Project appears.



There are three options in the PTR File menu.

If you choose Open from the menu, an information dialog box will appear either to indicate the working target project, i.e. the Current Project, if there is a Current Project;



or to remind you to activate a Power System Project if no Current Project exists (see Chapter 3 Managing projects). In the former case, you can redefine data structure for any elements of the Current Project through the PTR element buttons (in left column).

Redefining Busbar data structure


A Busbar button is provided for redefining busbar structure. Click the Busbar button and the data structure of the busbar parameter table will be given in the PTR window. You can define up to 80 fields for the busbar parameter table. In the table each field is composed of two descriptions:

- field-name;
- field-length.

The **field-name** must comply with the following rules:

- They can be up to 25 characters long.
- They can contain spaces, but cannot begin with a space.
- They can contain any printable character except double quotes("), brackets([]), braces({}), number signs(#), left or right parentheses(), and the character combination->.
- They cannot duplicate another field name in the same table.
- They should describe the contents of the field.

Power Shell supports a data type that is the alphanumeric field type permitting the full range of ASCII characters(except embedded null- ASCII 0), with a '**field-length**' not to exceed 25 characters.

In the busbar parameter table, the name of the first field cannot be changed by end-users. It is fixed and used by the Power Shell system.

After you redefine the data structure, you should save it by clicking Save in the PTR File menu. If you do not want to save the structure you have redefined, you can choose Cancel from the PTR File menu to ignore it.

Now you can open and redefine other element tables by clicking the relevant PTR element button.

Redefining Cable(Line) data structure

 A Cable button is provided for redefining cable(line) structure. Click the Cable button and the data structure of the cable(line) parameter table will be given in the PTR window. You can define up to 80 fields for the cable(line) parameter table. In the table, each field is composed of two descriptions:

- field-name;
- field-length.

The **field-name** must comply with the following rules:

- They can be up to 25 characters long.
- They can contain spaces, but cannot begin with a space.
- They can contain any printable character except double quotes("), brackets([]), braces({ }), number signs(#), left or right parentheses(), and the character combination->.
- They cannot duplicate another field name in the same table.
- They should describe the contents of the field.

Power Shell supports a data type that is the alphanumeric field type permitting the full range of ASCII characters(except embedded null- ASCII 0), with a '**field-length**' not to exceed 25 characters.

In the cable(line) parameter table, the name of the first field cannot be changed by end-users. It is fixed and used by the Power Shell system.

After you redefine the data structure, you should save it by clicking Save in the PTR File menu. If you do not want to save the structure you have redefined, you can choose Cancel from the PTR File menu to ignore it.

Now you can open and redefine other element tables by clicking the relevant PTR element button.

Redefining Load data structure


A Load button is provided for redefining load structure. Click the Load button and the data structure of the load parameter table will be given in the PTR window. You can define up to 80 fields for the load parameter table. In the table each field is composed of two descriptions:

- field-name;
- field-length.

The **field-name** must comply with the following rules:

- They can be up to 25 characters long.
- They can contain spaces, but cannot begin with a space.
- They can contain any printable character except double quotes("), brackets([]), braces({ }), number signs(#), left or right parentheses(), and the character combination->.
- They cannot duplicate another field name in the same table.
- They should describe the contents of the field.

Power Shell supports a data type that is the alphanumeric field type permitting the full range of ASCII characters(except embedded null- ASCII 0), with a '**field-length**' not to exceed 25 characters.

In the load parameter table, the name of the first field cannot be changed by end-users. It is fixed and used by the Power Shell system.

After you redefine the data structure, you should save it by clicking Save in the PTR File menu. If you do not want to save the structure you have redefined, you can choose Cancel from the PTR File menu to ignore it.

Now you can open and redefine other element tables by clicking the relevant PTR element button.

Redefining Transformer data structure

 A Transformer button is provided for redefining transformer structure. Click the Transformer button and the data structure of the transformer parameter table will be given in the PTR window. You can define up to 80 fields for the transformer parameter table. In the table each field is composed of two descriptions:

- field-name;
- field-length.

The **field-name** must comply with the following rules:

- They can be up to 25 characters long.
- They can contain spaces, but cannot begin with a space.
- They can contain any printable character except double quotes("), brackets([]), braces({}), number signs(#), left or right parentheses(), and the character combination->.
- They cannot duplicate another field name in the same table.
- They should describe the contents of the field.

Power Shell supports a data type that is the alphanumeric field type permitting the full range of ASCII characters(except embedded null- ASCII 0), with a '**field-length**' not to exceed 25 characters.

In the transformer parameter table, the name of the first field cannot be changed by end-users. It is fixed and used by the Power Shell system.

After you redefine the data structure, you should save it by clicking Save in the PTR File menu. If you do not want to save the structure you have redefined, you can choose Cancel from the PTR File menu to ignore it.

Now you can open and redefine other element tables by clicking the relevant PTR element button.

Redefining Generator data structure

 A Generator button is provided for redefining generator structure. Click the Generator button and the data structure of the generator parameter table will be given in the PTR window. You can define up to 80 fields for the generator parameter table. In the table each field is composed of two descriptions:

- field-name;
- field-length.

The **field-name** must comply with the following rules:

- They can be up to 25 characters long.
- They can contain spaces, but cannot begin with a space.
- They can contain any printable character except double quotes("), brackets([]), braces({}), number signs(#), left or right parentheses(), and the character combination->.
- They cannot duplicate another field name in the same table.
- They should describe the contents of the field.

Power Shell supports a data type that is the alphanumeric field type permitting the full range of ASCII characters(except embedded null- ASCII 0), with a '**field-length**' not to exceed 25 characters.

In the generator parameter table, the name of the first field cannot be changed by end-users. It is fixed and used by the Power Shell system.

After you redefine the data structure, you should save it by clicking Save in the PTR File menu. If you do not want to save the structure you have redefined, you can choose Cancel from the PTR File menu to ignore it.

Now you can open and redefine other element tables by clicking the relevant PTR element button.

Chapter 7 Copying project data

When you create a new Power System Project, you may use some data including the diagram of an existing Power System Project by copying the data and the diagram to your new Power System Project.

Copy data for a new Power System Project

To create a new Power System Project by copying data from an existing Power System Project, you must follow the steps below.

1. Create a new Power System Project

   From the Database menu in the Power Shell root window, choose the Create Project option, input a name for the Power System Project you want to create (see Input project name in chapter 5).

2. Redefine data structures for parameter tables if the required data structures are different from the default structures

   From the Database menu in the Power Shell root window, choose the Redefine Parameter option (see chapter 6 Redefining parameters).

3. Activate the Power System Project, from which you want copy the diagram and the data, as the Current Project

   From the File menu in the Power Shell root window, choose the Project option (see Activating a Power System Project in chapter 3).

4. Copy the diagram and the relative data of the Current Project to the new Power System Project

   From the Database menu in the Power Shell root window, choose the Copy Project option. The details are found in the section of How to copy in this chapter.

5. Input the rest of data for the new Power System Project when necessary

   From the File menu in the Power Shell root window, choose the Power System Diagram option (see Querying graphical element, and Input and update parameter in chapter 4).

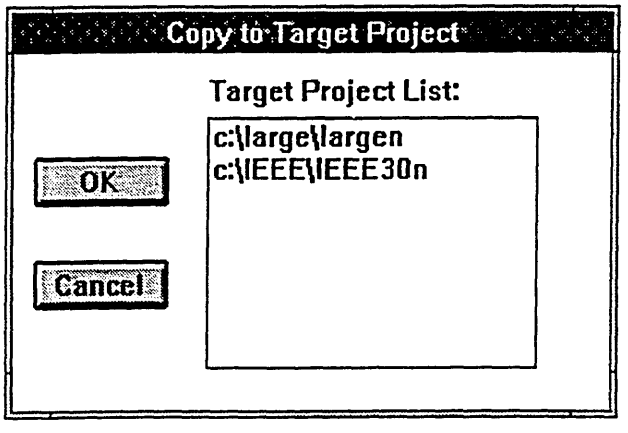
What will be copied

The Copy Project process will copy the Network configuration table and the Drawing Data table of the Current Project to the target Project, i.e. the Power System Project you have created and to which you want copy the data from the Current Project.

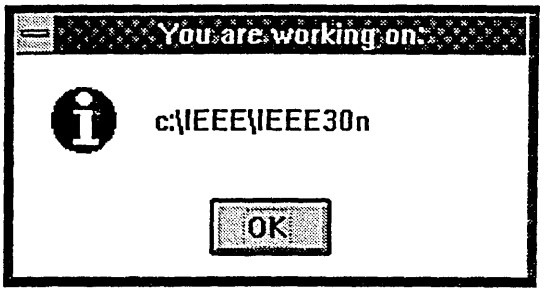
Also if any field of a data record of the Current Project has a description identical with that in the target Project, then the data of this field will be copied to the target Project. The rest of the data of the target Project should be input through data input procedure (see Querying graphical element, and Input and update parameter in chapter 4).

How to copy

When you choose the Copy Project option from the Database menu in the root window, a Copy to Target Project (CTP) window will appear.



Then an Information Dialog box will remind you of the Current Project.



After you click the OK button in the Information Dialog box, a list of the Power System Projects to which the Current Project can be copied will be given in the CTP Target Project List box. Now you have the following choices:

Copying Current Project to target Project

- Select the Power System Project, to which you want copy the Current Project data, from the Target Project List box.

- Choose the OK button in the CTP window.

Cancel Copy Project process

Before you choose the CTP OK button, you are always allowed to cancel the Copy Project process by choosing the Cancel button in the CTP window.

In the Copy Project process, first, any existing data of the target Project except the project name and the data structures will be deleted, and then the data of the Current Project will be copied to the target Project. Therefore you should start the Copy Project process just after you created the new Power System Project by giving the name and defining the data structures (either default or redefined), and before you input any other data to this Power System Project.

Part IIII Exchanging Data with Other Programs

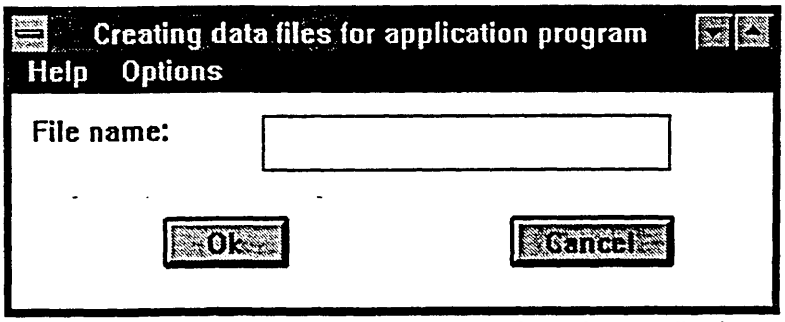
Chapter 8 Exporting data

Data Export, from the Tools menu in the Power Shell root window, provides a data exchange tool for the power system application programs that are not able to read the Power Shell data base table directly. Through Data Export you can transfer the Power System Project data to ASCII files. These ASCII files produced by Data Export contain information on the Power System Project, which by default includes:

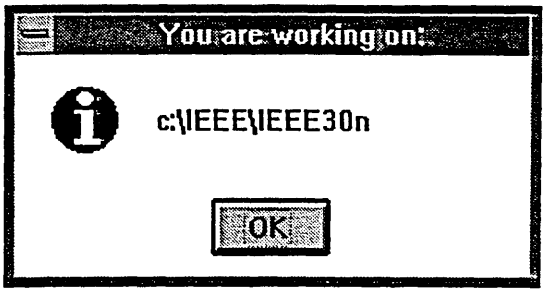
Total number of the records in the file;
Total number of the fields of each record;
Element identity;
Element class;
Element breaker status;
Element connection description (Input, Output);
Element name;
Element data (i), i=1,2..., $i \leq 79$.

The order of the element name and data in the ASCII files is the same as that in the Power System Project Parameter tables. The ascII files containing only a subset of the above information can also be produced when required.

When you choose the Data Export option from the Tools root menu, a Create Data File (CDF) window for data export will appear.



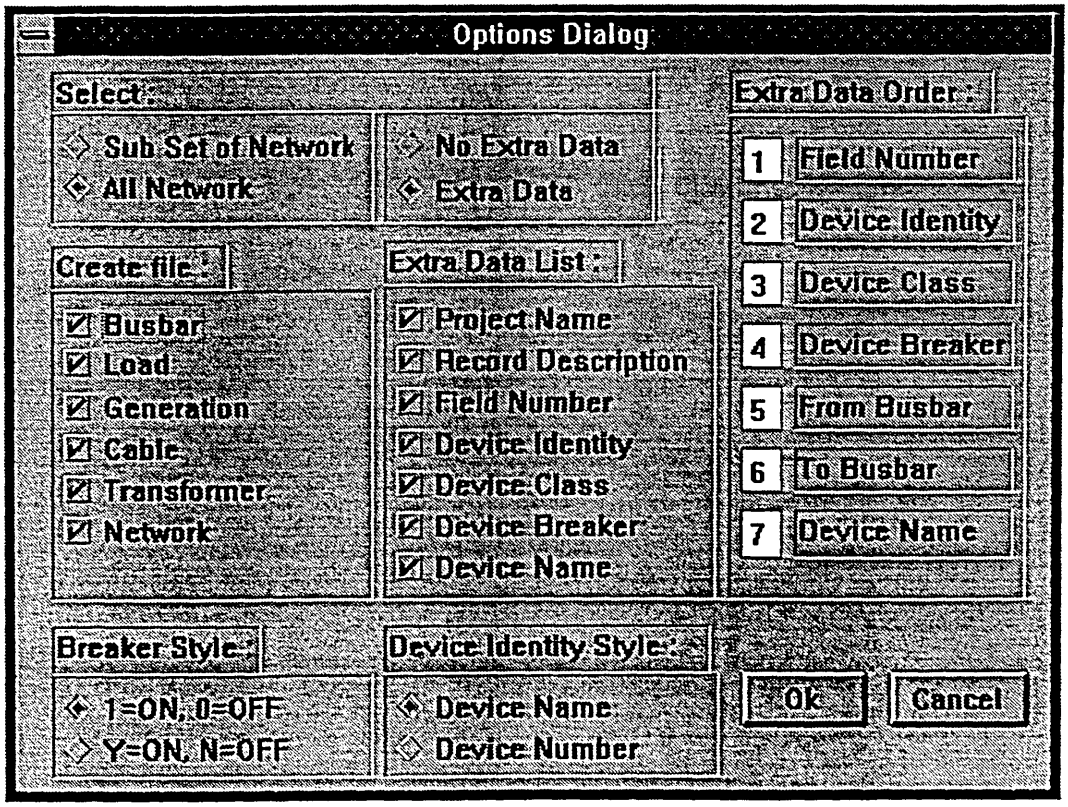
Then an Information Dialog box will remind you of the Current Project.



After you click the OK button in the dialog box, CDF will start to work on this Power System Project. Now you should input a file name into the File name box in the CDF window for the process of Data Export either by default or according to the items specified through the CDF Options menu.

Options in Data Export process

When you click Options in the CDF window, an Option Dialog box will appear.



You need to specify the Power System Project the data of which you want to export. Otherwise Data Export will produce the default ASCII files containing full information of the Current Project.

Select item

The Select section in the Option Dialog box contains two boxes. There are two options in the left box for selecting the covering area for Data Export.

1. Part of Network
   If you want to export data of part of the network which you have marked in the Drawing Power System Diagram window, you must choose the Part of Network option. Data Export will check whether the Part of Network option is selected before producing the ASCII files. If so the ASCII files containing the data of the elements you have marked will be produced.

2. Whole Network
   If you want to export data of the whole of the power system network, you must choose the Whole Network option. Data Export will check whether the Whole Network option is selected before producing the ASCII file. If so the ASCII files containing the whole of the network information will be produced.

In the right box of the Select section, two options are provided for specifying data requirements.

1. No Extra Data
   If you choose No Extra Data, the Data Export process will ignore all the Extra Data given in the Extra Data List box and produce the ASCII files containing the following information:

For Busbar,	Busbar IDn, Data(i), i=1,2,... and i≤79;
For Cable,	Input Busbar Name, Output Busbar Name , Data(i), i=1,2,... and i≤79;
For Transformer,	Input Busbar Name, Output Busbar Name , Data(i), i=1,2,... and i≤79;
For Load,	Input Busbar Name, Data(i), i=1,2,... and i≤79;
For Generator,	Output Busbar Nmae, Data(i), i=1,2,... and i≤79;

2. Extra Data

Besides the above information, Power Shell can also produce ASCII files containing some extra data that has been specified through the Extra Data option.

After you choose Extra Data from the right box of the Select section, you must specify the required extra data from the Extra Data List box.

You can mark or unmark any items in the Extra Data List to select or ignore it. The detailed description of the extra data is found in the section of Data structure of Data Export file in this Chapter.

Extra Data Order:
In the ASCII file produced by Data Export, each element data record is allowed to include the following fields:

Field Number
Device Identity
Device Breaker
From Busbar (except Busbar and Generator)
To Busbar (except Busbar and Load)
Device Name.

If you have requested the data record style with some of the above fields from the Extra Data List, Data Export will prefix the required extra data in the default order to each data record of the ASCII files. The default order is shown in number in the Extra Data Order box. If you would like to change the data order, you can change these order numbers, then Data Export will prefix the required extra data in your specified order.

Note: For busbar records, FromBusbar and ToBusbar are ignored automatically. For load records, ToBusbar is ignored, and for generator records, FromBusbar is ignored. In the ASCII files produced by Data Export, if any field of a record is ignored then the following fields will move forward sequentially.

Device Identity Style:
In the ASCII file produced by Data Export, each element data record may or may not contain the Identity field. If you have requested the data record style with Identity field from the Extra Data List, Data Export will use 'Device Name' as the default style of the Identity. If you would like to change the Identity style, e.g. Number has been used as the device identity in PSS/E, you can choose options from the Device Identity Style box, then Data Export will describe element identities in the selected style.

Breaker Style:
In the ASCII file produced by Data Export, each element data record may or may not contain the Breaker field, which is for specifying the ON or OFF status of a breaker. If you have requested the data record style with Breaker field from the Extra Data List, Data Export will use '1' for breaker ON and '0' for breaker OFF as the default style. If you would like to change the Breaker style, e.g. Y or N has been used to describe whether a element is or is not in operation in POSCODAM of ABB, you can choose options from the Breaker Style box, then Data Export will describe breakers in the selected style.

Note: If you want Power Shell to read back the ASCII files produced by your application program, you must choose Extra Data from the right box of the Select section in the Option Dialog box. And Device Name and Device Class must be selected and to be used in generating the calculation result ASCII files of your application program as the index key for the Power Shell Data Read Back process.

Create File

Data Export can produce you categorised data export files. If your disk has only a limited space, you can unmark a certain category of the power system elements, e.g. Busbar, from the list in the Create File box. Then Data Export will ignore the data files of the unmarked elements when producing ASCII files.

Leaving Option Dialog box

When you leave the Option Dialog box, you should click the OK button to confirm your selections in the Option Dialog box, or click the Cancel button to cancel any selections you have done.

Input a legal file name

CDF File-name box

In order to produce the ASCII data files in the director you designated, you must enter a legal file name in the Create Data File(CDF) File-name box. A legal file name takes the form of:

C:\PathName\FileName.TXT

where

PathName is up to eight characters;
FileName is up to six characters;

TXT is the required extension name.

If in the Option Dialog box you required categorised data files, the names of these categorised data files will be given automatically by the Data Export program in the following form:

C:\PathName\FileNameN.TXT	Network configuration file
C:\PathName\FileNameC.TXT	Cable(Line) file
C:\PathName\FileNameB.TXT	Busbar file
C:\PathName\FileNameG.TXT	Generator file
C:\PathName\FileNameL.TXT	Load file
C:\PathName\FileNameT.TXT	Transformer file

Leaving CDF

After you enter a legal file name, you can click the CDF OK button to start producing ASCII files. Then you can obtain these ASCII files in the PathName directory.

Before you click the CDF OK button you are always allowed to cancel any procedures and close CDF by clicking the CDF Cancel button. If you do so, no ASCII files will be produced.

Data structure of Data Export files

The ASCII files produced by Data Export take the following form.

Integrated Data Export ASCII files

Power System Project name: DriveName\PathName\PjName

where

PathName is composed of up to 8 ASCII characters;
PjName is the Project name and is composed of up to 7 ASCII characters.

Record description: [RN][BRN][CRN][TRN][LRN][GRN]

where

RN: the total number of the records in the file;
BRN: the total number of Busbar records;
CRN: the total number of Cable(Line) records;
TRN: the total number of Transformer records;
LRN: the total number of Load records;
GRN: the total number of Generator records.

Busbar data: [FieldN][IDn][Dclass][Breaker][Name][Data(i)]

where

FieldN: the total number of the fields in each Busbar record;
IDn: the identity number of the busbar, i.e. the record number in the database;
Dclass: B, the device class for busbars;
Breaker: 0, for breaker off; or 1, for breaker on;
Name: the busbar name;
Data(i): busbar data or parameter in a structure identical with that in the database; i=1,2,...and i≤79.

Cable(Line) data:

[FieldN][IDn][Dclass][Breaker][Input][Output][Name][Data(i)]

where

FieldN: the total number of the fields in each Cable(Line) record;
IDn: the identity number of the cable(line), i.e. the record number in the database;
Dclass: C, the device class for Cables(Lines);
Breaker: 0, for breaker off; or 1, for breaker on;
Input: busbar name;
Output: busbar name;
Name: the cable(line) name;
Data(i): cable(line) data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

Transformer data:

[FieldN][IDn][Dclass][Breaker][Input][Output][Name] [Data(i)]

where

FieldN: the total number of the fields in each Transformer record;
IDn: the identity number of the transformer, i.e. the record number in the database;
Dclass: T, the device class for transformers;
Breaker: 0, for breaker off; or 1, for breaker on;
Input: busbar name;
Output: busbar name;
Name: the transformer name;
Data(i): transformer data or parameter in a structure identical with that in the database; i=1,2,... and i≤ 79.

Load data: [FieldN][IDn][Dclass][Breaker][Input][Name] [Data(i)]

where

FieldN: the total number of the fields in each Load record;
IDn: the identity number of the load, i.e. the record number in the database;
Dclass: L, the device class for loads;
Breaker: 0, for breaker off; or 1, for breaker on;
Input: busbar nmae;
Name: the load name;
Data(i): load data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

Generator data: [FieldN][IDn][Dclass][Breaker][Output][Name] [Data(i)]

where

FieldN: the total number of the fields in each Generator record;
IDn: the identity number of the generator, i.e. the record number in the database;
Dclass: L, the device class for generators;
Breaker: 0, for breaker off; or 1, for breaker on;
Output: busbar name;
Name: the generator name;
Data(i): generator data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

Categorised Data Export ASCII files

The data structure of Data Export ASCII files for the individual category of the power system elements, i.e. busbars, cables, transformers, loads and generators, is identical with that in the corresponding sections of the integrated data export ASCII file. These are detailed as follows.

1. Data structure of Busbar data export ASCII files:

[FieldN][IDn][Dclass][Breaker][Name][Data(i)]

where

FieldN: the total number of the fields in each Busbar record;
IDn: the identity number of the busbar, i.e. the record number in the database;
Dclass: B, the device class for busbars;
Breaker: 0, for breaker off; or 1, for breaker on;
Name: the busbar name;
Data(i): busbar data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

2. Data structure of Cable(Line) data export ASCII files:

[FieldN][IDn][Dclass][Breaker][Input][Output][Name][Data(i)]

where

FieldN: the total number of the fields in each Cable(Line) record;
IDn: the identity number of the cable(line), i.e. the record number in the database;
Dclass: C, the device class for Cables(Lines);
Breaker: 0, for breaker off; or 1, for breaker on;
Input: busbar nmae;
Output: busbar name;
Name: the cable(line) name;
Data(i): cable(line) data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

3. Data structure of Transformer data export ASCII files:

[FieldN][IDn][Dclass][Breaker][Input][Output][Name] [Data(i)]

where FieldN: the total number of the fields in each Transformer record;
IDn: the identity number of the transformer, i.e. the record number in the database;
Dclass: T, the device class for transformers;
Breaker: 0, for breaker off; or 1, for breaker on;
Input: busbar name;
Output: busbar name;
Name: the transformer name;
Data(i): transformer data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

4. Data structure of Load data export ASCII files:

[FieldN][IDn][Dclass][Breaker][Input][Name] [Data(i)]

where FieldN: the total number of the fields in each Load record;
IDn: the identity number of the load, i.e. the record number in the database;
Dclass: L, the device class for loads;
Breaker: 0, for breaker off; or 1, for breaker on;
Input: busbar name;
Name: the load name;
Data(i): load data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

5. Data structure of Generator data export ASCII files:

[FieldN][IDn][Dclass][Breaker][Output][Name] [Data(i)]

where FieldN: the total number of the fields in each Generator record;
IDn: the identity number of the generator, i.e. the record number in the database;
Dclass: L, the device class for generators;
Breaker: 0, for breaker off; or 1, for breaker on;
Output: busbar name;
Name: the generator name;
Data(i): generator data or parameter in a structure identical with that in the database; i=1,2,... and i≤79.

An example

c:\ieeef\ieeef.txt - An example of integrated data export ASCII files

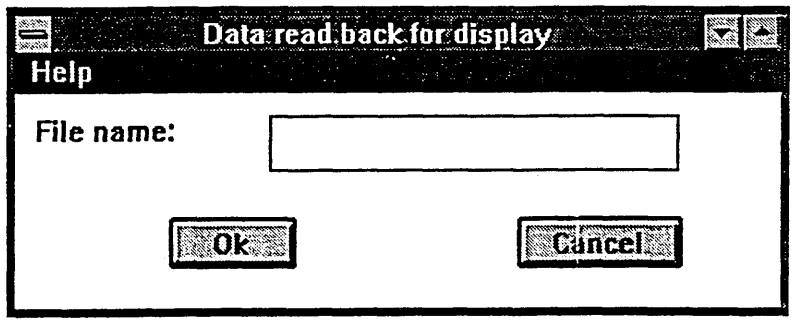
```
                Power System Project Name
c:\IEEE\IEEE30 ↵
                              Record Description
97  30  34  7  20   6 ↵
 \   \   \   \   \   \
[RN][BRN][CRN][TRN][LRN][GRN]
                                                                  ↙ Busbar data
                 [Dclass]
[FieldN][IDn]  I [Breaker][Name][Data(1)][Data(2)]          .....                    [Data(15)]
 /    /    /   /     /     \     /                                                    /
20  11  B  0  SS19  300  0  0  0  0  0  0  0  0  0  0  0  0  0  0
20  17  B  1  SS24  300  0  0  0  0  0  0  0  0  0  0  0  0  0  0
.....
20  28  B  0  SS7   175  0  0  0  0  0  0  0  0  0  0  0  0  0  0
                                                                  ↙ Cable(Line) data
                 [Dclass]
[FieldN][IDn]  I [Breaker] [Input]  [Output] [Name]   [Data(1)][Data(2)]    .....    [Data(7)]
 /    /    /   /       /       /       \      \    /                              /
14  11  C  1  SS18    SS19    TL23  0  0  0  0  0  0  0
......
14  34  C  1  SS5     SS7     TL8   0  0  0  0  0  0  0
                                                                  ↙ Transformer data
                 [Dclass]
[FieldN][IDn]  I [Breaker] [Input][Output][Name][Data(1)][Data(2)]                .....                [Data(15 )]
 /    /    /   /       /       /       \      \    /                                                  /
22   5  T  1  SS4     SS12    TL15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
22   6  T  1  SS13    SS12    TL16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
......
22   1  T  1  SS6     SS9     TL11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
                                                                  ↙ Load data
                 [Dclass]
[FieldN][IDn]  I [Breaker] [Input]  [Name][Data(1)][Data(2)] ..... [Data(5)]
 /    /    /   /       /       /       \      \    /                  /
11   7   L  1  SS19    LD16  0  0  0  0  0
......
11  19  L  1  SS7     LD7   0  0  0  0  0
                                                                  ↙ Generator data
                 [Dclass]
[FieldN][IDn]  I [Breaker][Output] [Name]    [Data(1)][Data(2)] ..... [Data(7)]
 /    /    /   /       /       /       /      /                              /
13   6   G  1  SS13    GEN6  0  0  0  0  0  0  0
......
13   3   G  1  SS5     GEN3  0  0  0  0  0  0  0
```

Note: The above example is given in the default Data Export form, i.e. all the extra data is selected.

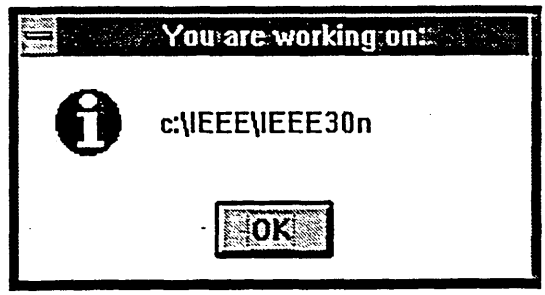
Chapter 9 Importing data

The Data Read Back program can read the export ASCII data files of user application programs, write the data to the data base table of Power Shell, and display the data on the screen diagram. Also the read-back data of a power system element can indicate the element current status by keeping or changing its display style in the screen diagram. The read back data could be from any application programs such as simulation, planning and on-line data acquisition programs. The only condition for the Data Read Back program is that the data produced by these application programs must be in the structure designated by Power Shell, and extra data Device Name and Device Class must be selected in the Data Export process (see the section of Options in Data Export process in chapter 8 Exporting Data)

When you choose Data Read Back from the Tools menu in the Power Shell root window, a Data Read Back (DRB) window will appear.



Then an Information Dialog box will remind you of the current Power System Project.



After you click the OK button in the dialog box , DRB will start to work on this Power System Project. Now you should input a fiel name into the File Name box in the DRB window for the Data Read Back process.

Input a legal file name

You must enter a legal file name to the DRB File-name box. This file will be read and written to the database table of the Current Project as the source file. This source file must be an ASCII file and in the structure designated by Power Shell. A legal file name takes the form of:

C:\PathName\FileName.TXT

where

PathName is up to 8 characters
FileName is up to 6 characters

After you enter a legal file name, you can click the DRB OK button to start reading and writing the data to the database table. When the Data Read Back procedure has finished, you can view the data in the Drawing Power System Diagram window.

Before you click the DRB OK button, you are always allowed to cancel any procedures and close the DRB window by clicking the DRB Cancel button. If you do so, no data will be written to the database tables of you Power System Project.

Data structure of Data Read Back ASCII files

[RN] where RN is the total number of the records in the file.

[Dclass][Name][IC+Key][Information]

where

Dclass: Device Class, which could be
B or b, for Busbars;
C or c, for Cables(Lines);
T or t, for Transformers;
L or l, for Loads;
G or g, for Generators;

Name: Device Name that is composed of up to ten ASCII characters;

IC: Information Class, which could be anyone of the twenty six English characters in either case, e.g. U, I, p, g, D, R, ...

Key: Device running status key, which could be:
O or o, for 'over' status;
L or l, for ' low' status;
=, for 'normal' status.
The Power Shell system may change the power system element display style in the diagram after reading this key.

Information: Information from users application programs, e.g. the results of calculations, which is composed of up to 13 ASCII characters.

An example

c:\ieeef\databack.txt - An example of data read back ASCII files

```
[RN]
 /
95
[Dclass][Name][IC+Key][Information]
 /     /     /      /
B  SS19  V=   300
B  SS23  VL  299.8
B  SS29  Vo  300.5
B  SS30  VO  300.8
......
B  SS7  VL  173.492
C  TL23  pq  0.0575J0.3214
C  TL30  X=  0.18520
C  TL23  PQ  0.1737+0.2541
......
C  TL8  X=  0.18520
T  TL16  P=  300
T  TL12  PL  500
T  TL14  Po  320
......
T  TL11  P=  130
L  LD16  D=  0.02
L  LD20  D=  0.00
......
L  LD7  D=  0.09
G  GEN6  D=  0.180
......
G  GEN3  D=  0.03
```

Appendix A PowerApp
- An example for other C programs to read/write ASCII files from/to Power Shell

PowerApp written in C is a program for reading the integrated ASCII files produced by Power Shell into the Read-in data buffer of your calculation program written in C, and writing the data from the Calculation-result data buffer of your calculation program to the ASCII file that can be read back by Power Shell to its database. And also this example provides your calculation program with a simple interface under MS-Windows, i.e. you can now run your program in MS-Windows environment if it was a Non-Windows applications.

The PowerApp program is composed of the following components.

1. PowerApp.prj - project file
2. PowerApp.c - source code
3. PowerApp.rc - resource file required for programming in MS-Windows
4. PowerApp.def - definition file required for programming in MS-Windows
5. Power.ico - icon file required for programming in MS-Windows
6. PowerApp.exe - executable code

With PowerApp, to read/write ASCII files from/to Power Shell, all you require is to calculate the data in the DataBuff and write the results to the KeyBuff in PowerApp.c. To do so you can simply embed your calculation program as the sub routine to the function of PowerSystemCalculation() in PowerApp.c. Then you can open c:\powerapp\PowerApp.prj to compile PowerApp.prj in the environment of Borland/Turbo C 3.0 or later. Before you compile PowerApp.prj, you must notify the sort of the executable code to be produced by choosing the option of Windows-App from the Application submenu in the Options menu.

If your calculation program requires a data structure different from the data structure given in PowerApp.c, you can update the data structure in PowerApp.c. If you just updated the data structure and/or embedded your calculation program, then you needn't have to know anything about programming in MS-Windows. If you wish to change the display style of the PowerApp program, then you need to have some knowledge of programming in MS-Windows.

The PowerApp program is found in c:\powerapp after you install Power Shell.

Appendix B Re-creating IEEE30 for PSS/E
- An example for transferring Power System Project format

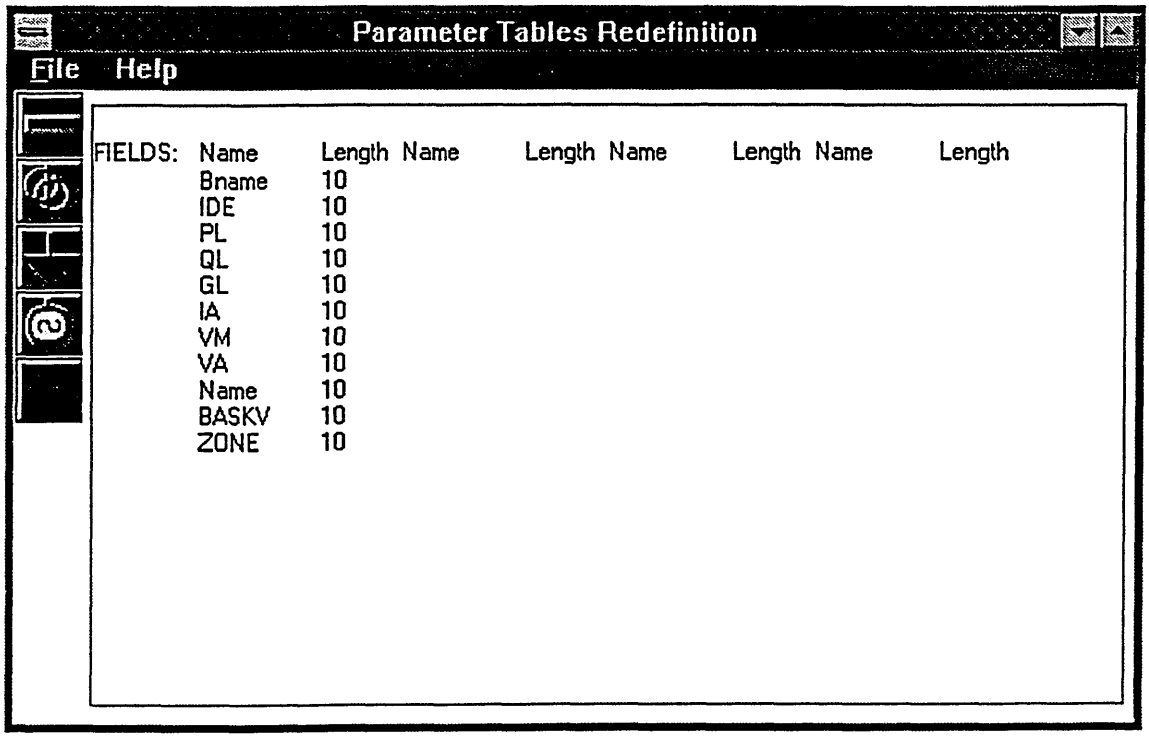
Currently there are two example Power System Projects in Power Shell, i.e. IEEE30 and LARGE. You can re-create any of them (actually any existing Power System Projects, if you have created more) for the load flow format required. The following steps give an example of re-creating IEEE30 for the PSS/E data format.

1. Create Power System Project PSSE
Choose the **Create Project** option from the **Database** menu in the Power Shell root window, then input C:\PSSE\PSSE as the Project name.

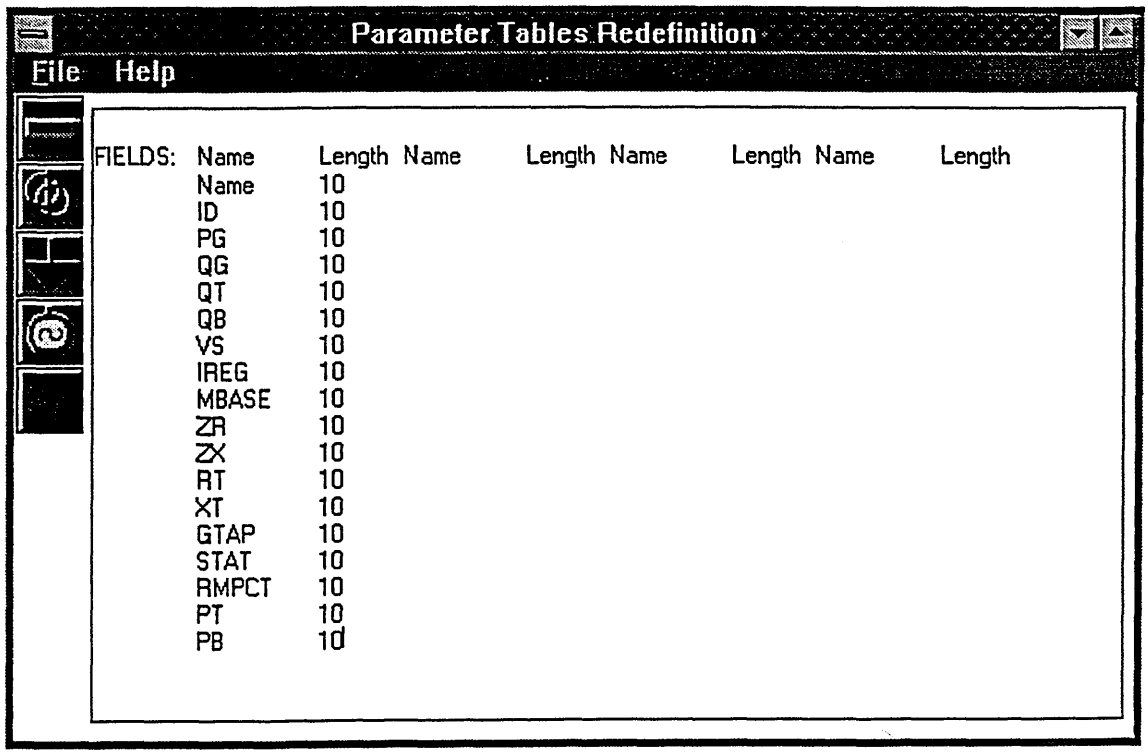
2. Activate C:\PSSE\PSSE
Choose the **Project** option from the **File** menu in the Power Shell root window, and then select C:\PSSE\PSSEN and activate it as the current Project.

3. Redefine parameter table structure

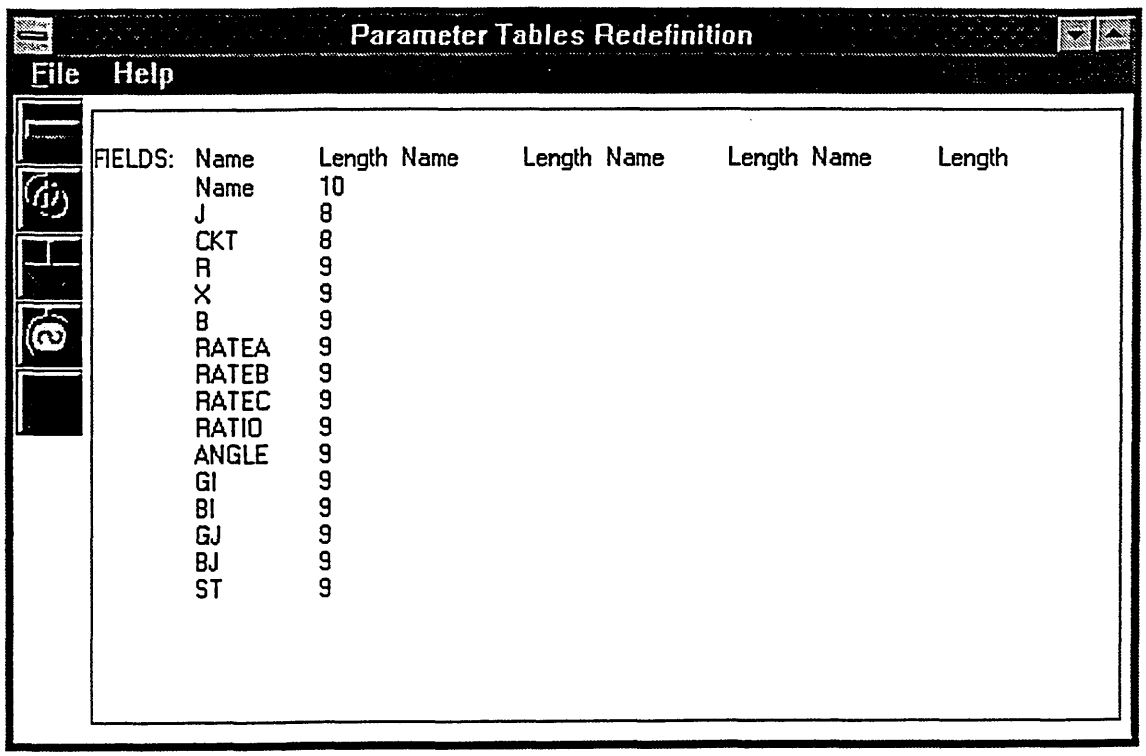
For Busbar, update the original IEEE30 data format into the PSS/E data format:



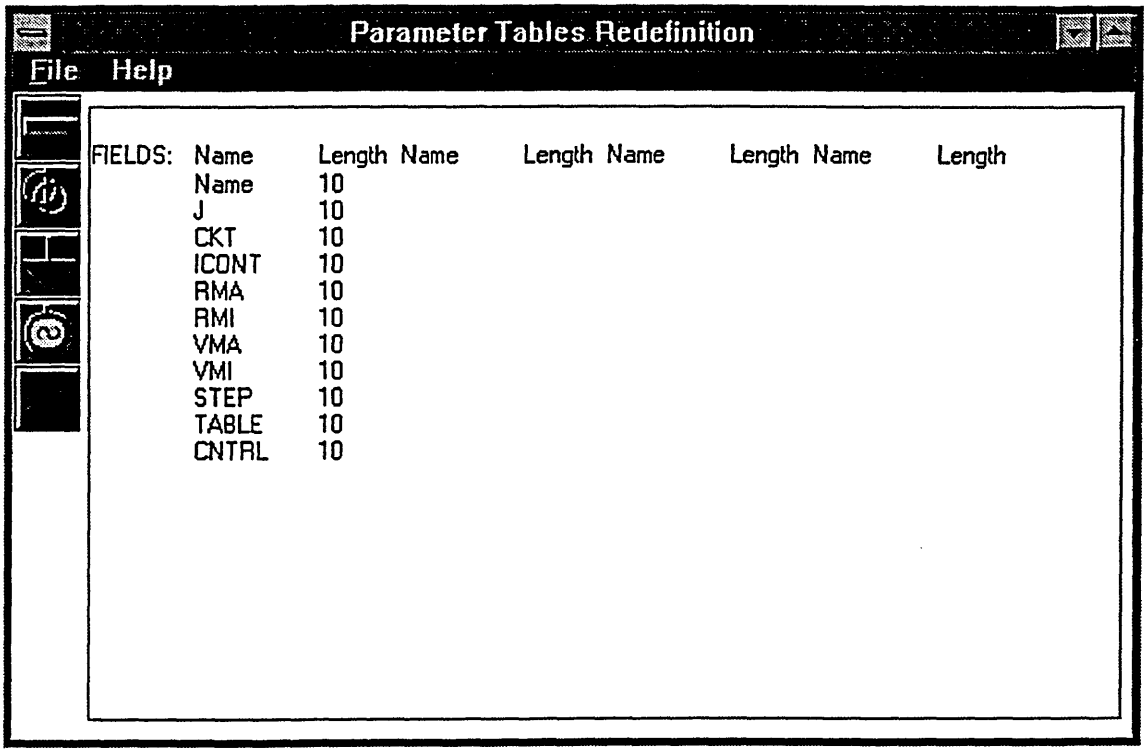
For Generator, update the original IEEE30 data format into PSS/E data format:



For Cable (Branch), update the original IEEE30 data format into PSS/E data format:



For Transformer, update the original IEEE30 data format into PSS/E data format:



4. Copy IEEE30 to PSSE

- Choose the **Project** option from the **File** menu in the Power Shell root window, and then select C:\IEEE\IEEE30N and activate it as the source file for copy process.
- Choose the **Copy Project** option from the **Database** menu in the Power Shell root window, and then
  * select C:\PSSE\PSSEN as the target Project for Copy process
  * select OK button to start Copy

5. Update element parameters for PSSEN

- Choose the **Project** option from the **File** menu in the Power Shell root window, and then activate C:\PSSE\PSSEN as the Current Project.
- Choose the **Power System Diagram** option from the **File** menu in the Power Shell root window, and then input parameters for the elements of PSSEN (see Querying graphical element, and Input and update parameter in Chapter 4, User's Guide)

  Note, to update elements at the Middle Voltage level, you need to Zoom In the diagram (see Middle-layer of the Drawing board in Chapter 4, User's Guide)

6. Produce ASCII file for PSSEN

- Choose the **Data Export** option from the **Tools** menu in the Power Shell root window
- Choose the **No Extra Data** option in the **Options** box
- Input the ASCII file name for target Project PSSEN, e.g. C:\PSSEF\PSSEF.TXT
- Choose OK to start Data Export

7. Access to PSSEF files

Using File Manager in MS-Windows to open PSSEF.TXT, PSSEFB.TXT and
PSSEFC.TXT.